Saarland University

Faculty of Natural Sciences and Technology I

Department of Computer Science

SELECTED TOPICS IN ALGORITHMIC GEOMETRY

Dissertation
zur Erlangung des Grades
des Doktors der Ingenieurwissenschaften (Dr.-Ing.)
der Naturwissenschaftlich-Technischen Fakultät I
der Universität des Saarlandes

Victor Manuel Alvarez Amaya

Saarbrücken, 2012

Dean of the Faculty of Natural Sciences and Technology I at Saarland University	Prof. Dr. Mark Groves
Colloquium	20 December 2012
<u>Examination Board</u>	
Supervisor and first reviewer	Prof. Dr. Raimund Seidel Chair of Theoretical Computer Science Saarland University Saarbrücken, Germany
Second reviewer	Prof. Dr. Oswin Aichholzer Institute for Software Technology Graz University of Technology Graz, Austria
Chairman	Prof. Dr. Dr. h.c. mult. Reinhard Wilhelm Chair for Programming Languages and Compiler Construction Saarland University Saarbrücken, Germany
Research Assistant	Dr. Tobias Mömke Chair of Computational Complexity Saarland University Saarbrücken, Germany

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus anderen Quellen oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet.

Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder ähnlicher Form in einem Verfahren zur Erlangung eines akademischen Grades vorgelegt.

Saarbrücken, 26. September 2012

Victor Manuel Alvarez Amaya

ABSTRACT

Let $P \subset \mathbb{R}^2$ be a set of n points with no three points on a line. A crossing-free structure on P is a straight-edge plane graph whose vertex set is P.

In this thesis we consider problems of two different topics in the area of algorithmic geometry: Geometry using Steiner points, and counting algorithms. These topics have P and certain crossing-free structures on P as our primary objects of study. Our results can be roughly described as follows:

- Given a k-coloring of P, with $k \geqslant 3$ colors, we will show how to construct a set of Steiner points $S = S(P)$ such that a k-colored quadrangulation can always be constructed on $P \cup S$. The bound we show of $|S|$ significantly improves on previously known results.

- We also show how to construct a set $S = S(P)$ of Steiner points such that a triangulation of $P \cup S$ having all its vertices of even (odd) degree can always be constructed. We show that $|S| \leqslant \frac{n}{3} + c$, where c is a constant. We also look at other variants of this problem.

- With respect to counting algorithms, we show new algorithms for counting triangulations, pseudo-triangulations, crossing-free matchings and crossing-free spanning cycles on P. Our algorithms are simple and allow good analysis of their running times. These algorithms significantly improve over previously known results. We also show an algorithm that counts triangulations approximately, and a hardness result of a particular instance of the problem of counting triangulations exactly.

- We show experiments comparing our algorithms for counting triangulations with another well-known algorithm that is supposed to be very fast in practice.

ZUSAMMENFASSUNG

Sei P eine Menge von n Punkte in der Ebene, so dass keine drei Punkten auf einer Geraden liegen. Eine kreuzungsfreie Struktur von P ist ein geradliniger ebener Graph, der P als Knotenmenge hat.

In dieser Dissertation behandeln wir zwei verschiedene Problemkreise auf dem Gebiet der algorithmischen Geometrie: Geometrie mit Steinerpunkten und Anzahl bestimmende Algorithmen auf P und auf gewissenen kreuzungsfreien Strukturen von P. Unsere Resultate können wie folgt beschrieben werden:

- Gegeben sei eine k-Färbung von P, mit $k \geqslant 3$ Farben. Es wird gezeigt, wie eine Menge $S = S(P)$ von Steiner Punkten konstruiert werden kann, die die Konstruktion einer k-gefärbten Quadrangulierung von $P \cup S$ ermöglicht. Die von uns gezeigte Schranke für $|S|$ verbessert die bisher bekannte Schranke.

- Gezeigt wird auch die Konstruktion einer Menge $S = S(P)$ von Steiner Punkten, so dass eine Triangulierung von $P \cup S$ konstruiert werden kann, bei der der Grad aller Knoten gerade (ungerade) ist. Wir zeigen, dass $|S| \leqslant \frac{n}{3} + c$ möglich ist, wobei c eine Konstante ist. Wir betrachten auch andere Varianten dieses Problems.

- Was die Anzahl bestimmenden Algorithmen betrifft, zeigen wir neue Algorithmen, um Triangulierungen, Pseudotriangulierungen, kreuzungsfreie Matchings und kreuzungsfreie aufspannende Zyklen von P zu zählen. Unsere Algorithmen sind einfach und lassen eine gute Analyse der Laufzeiten zu. Diese neue Algorithmen verbessern wesentlich die bisherigen Ergebnisse. Weiter zeigen wir einen Algorithmus, der Triangulierungen approximativ zählt, und bestimmen die Komplexitätsklasse einer bestimmten Variante des Problems des exakten Zählens von Triangulierungen.

- Wir zeigen Experimente, die unsere triangulierungszählenden Algorithmen mit einem anderen bekannten Algorithmus vergleichen, der in der Praxis als besonders schnell bekannt ist.

ACKNOWLEDGMENT

I am profoundly grateful to Raimund Seidel for the time he devoted to me, for his support, and for his patience while doing research together. His comments and suggestions always enriched my knowledge and made this work better. I am indebted to him for giving me the chance to find my own path.

I would like to thank Oswin Aichholzer for agreeing to be on my thesis committee and for the wonderful conversations about the topics presented in this thesis that we had the chance to have from time to time.

Many of the results found in this thesis are product of the work I did with Karl Bringmann, Radu Curticapean, and Saurabh Ray. I would like to thank them for the amazing time we had while doing research on triangulations. I learned a lot from them.

I would also like to thank the Deutsche Forschungsgemeinschaft (DFG) for the scholarship I received during the first two-and-a-half years of my PhD.

Finally, I would like to acknowledge additional financial support from CONACYT-DAAD of México.

To my family and friends.

To Juliane.

CONTENTS

CHAPTER 1

INTRODUCTION

Combinatorial geometry deals with problems that consists of geometric entities, such as points, lines, polygons, etc., and studies structural problems defined on those geometric entities. Many problems in combinatorial geometry tend to look very innocent, yet they usually are extremely hard. For example, the following innocent-looking question is one of the iconic problems in combinatorial geometry: How many non-equivalent configurations of points are there on the plane? Probably, in order to make more sense out of this question we have to explain what do we mean by "non-equivalent".

Given three points $p, q, r \in \mathbb{R}^2$, we say that $p \to q \to r$ do a right turn if the determinant of the matrix $\begin{pmatrix} r_x - p_x & q_x - p_x \\ r_y - p_y & q_y - p_y \end{pmatrix}$ is positive. They do a left turn if the determinant is negative, and p, q, r are collinear if the determinant is zero, *i.e.* no turn. See Figure 1.1.

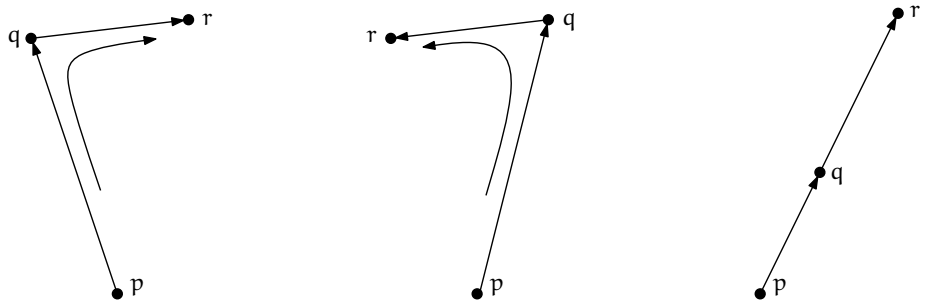

Figure 1.1 – To the left a right turn. In the middle a left turn. To the right no turn.

We define the *orientation* of the triple $p \to q \to r$ as the value of their determinant, as explain above. Two sets $P, Q \subset \mathbb{R}^2$ of $n \geqslant 3$ points are equivalent if there is a bijective map between P and Q such that the orientation of *all* their triples of points coincide. In such a case we will say that P and Q have the same *order type*. Thus, we say that P, Q are *non-equivalent* if and only if they have different order types. So, for example, there is *exactly one* set of three points on the plane, see to the left in Figure 1.1, there are *exactly two* non-equivalent sets of four points, see Figure 1.2, and in general, there are $2^{\Theta(n \log(n))}$ non-equivalent sets of n points on the plane. So there is a satisfying answer to the question posed above.[I] The answer, however, did not come as easily as we might have expected in the beginning. The question was posed by Jacob E. Goodman and Richard Pollack in [39] in '83, although not in exactly this form, and the satisfying answer came years later after intensive highly non-trivial research, see [41, 40] and references therein.

Order types are still nowadays a very active topic of research. The literature on order types is however, and unfortunately, very scattered. We strongly recommend the literature by J. E. Goodman and R. Pollack, who introduce order types in first place, and the literature by Oswin Aichholzer, Franz Aurenhammer, and Hannes Krasser, who have done extensive research on order types in the recent years, see [50] for example.

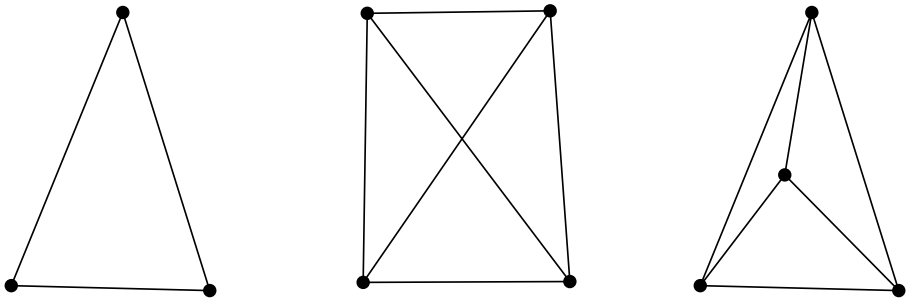

Figure 1.2 – With three points there is exactly one order type (left). With four points there are exactly two (middle, right). The shown line segments are all the ones that can be found on each set having endpoints at points of the sets.

Order types are the cornerstones of many problems in combinatorial geometry. Let us for example ask the following: Given a set $P \subset \mathbb{R}^2$ of $n \geqslant 3$ points, how many different maximal cardinality sets of pairwise non-crossing[II] straight-line segments among the elements of P are there? See Figure 1.3 for an example of what two different such sets look like. It is clear that whether two straight-line segments intersect depends solely on

[I] Where satisfying means that up to multiplicative constants in the exponent, the bound on the number of non-equivalent sets of points on the plane is tight.

[II] Two straight-line segments are called crossing if and only if their intersection point lies in the strict interior of at least one of them.

the order type of their endpoints. If they look like in the middle in Figure 1.2, they intersect if and only if they are the diagonals of the quadrilateral. If they look like the configuration to the right in Figure 1.2, they never intersect. Thus the number of such sets of maximal cardinality[III] depends only on the order type of P. Sets of points of the same order type will always give the same answer, while sets of different order types will give in general different answers.

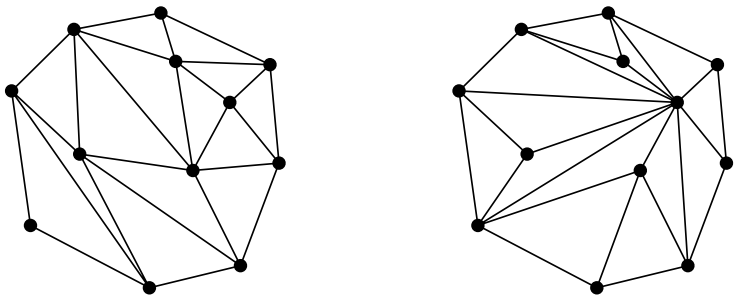

Figure 1.3 – Two different "triangulations" of the same set of points.

The previous question might also look innocent, but as it turns out, no satisfying answer has been found to this day, although this problem has been the subject of intensive research in the last 30 years. It is nevertheless known today that the answer lies somewhere between $\Omega(2.4^n)$ and $O(30^n)$, see [77, 75]. Making the gap tight is what has cost so far a lot of effort. It is widely believed that the tight answer for the lower end should be $\Omega\left(\sqrt{12}^n\right) \approx \Omega(3.464^n)$. In [72] the configuration that achieves this number is presented. It is also believed that the tight answer for the high end should be way smaller, probably $O(15^n)$, or even $O(10^n)$. A configuration having $\Omega(8.65^n)$ is currently known [33]. This is an extremely challenging problem.

The problem of counting the "triangulations" of P comes also in another flavor. If P is given, we currently know that its number of triangulations lies somewhere between $\Omega(2.4^n)$ and $O(30^n)$. Now imagine that we are actually interested in the *exact number* produced by P. How do we compute such a number? Well, it turns out that this is also a very challenging problem that is still not well-understood, and whose background is also more recent than that of the original question. This variant asks essentially for algorithmic techniques that can exploit the structure of P to compute such a number.[IV] Thus, properly, we enter the realm of what is known as *algorithmic geometry*.

[III] These sets are commonly known as triangulations of P. A proper definition will be given later on in the chapter.

[IV] Since a formula seems in general out of reach.

The difference between combinatorial geometry and algorithmic geometry could really be a thin line, but we can safely say that algorithmic geometry is the algorithmic part of combinatorial geometry.

In this thesis we give our small contribution to the area of algorithmic geometry. We show results in different subjects of the area, one of them being the algorithmic version of the problem of counting "triangulations" just explained.

Our thesis contains most of our research work between the end of 2007 and beginning of 2012. Part of this work has been presented at conferences, mostly in preliminary forms, see [8, 10, 9].

For excellent, and general, references on the areas of combinatorial and algorithmic geometry, we encourage the reader to take a look at the following books: Algorithms in Combinatorial Geometry by Herbert Edelsbrunner, see [34]. Combinatorial Geometry by János Pach and Pankaj K. Agarwal, see [63]. Computational Geometry – An Introduction by Franco P. Preparata and Michael I. Shamos, see [67]. Algorithmic Geometry by Jean-Daniel Boissonnat and Mariette Yvinec, see [18]. Computational Geometry: Algorithms and Applications by Mark de Berg, Marc van Kreveld, Mark Overmars and Otfried Schwarzkopf, see [27].

With respect to the way the thesis is written, we have chosen to write each chapter as self-contained as possible. This means that each chapter contains thorough introductions about the topic therein studied, and also its conclusions. All thesis-wide definitions and notation are stated in this chapter. Thus, upon reading this chapter, the reader should feel free to jump to any other chapter as he/she sees fit.

We have divided the rest of the chapter as follows: In § 1.1 we define the basic concepts that we will use throughout this work. In § 1.2 and § 1.3 we state the topics and results found in this work. Finally, in § 1.4 we briefly explain the model of computation we work on.

1.1 Thesis-wide definitions

Definition 1.1 (Simple polygon). A simple polygon $\mathcal{P} \subset \mathbb{R}^2$ is a single closed polygonal chain that does not intersect itself.

An example of a simple polygon can be seen to the left in Figure 1.4. Example of objects that are no simple polygons can also be seen in Figure 1.4.

The corners of a polygon $\mathcal{P}$ are called its vertices, and the straight-line segments connecting vertices are called the edges of $\mathcal{P}$.

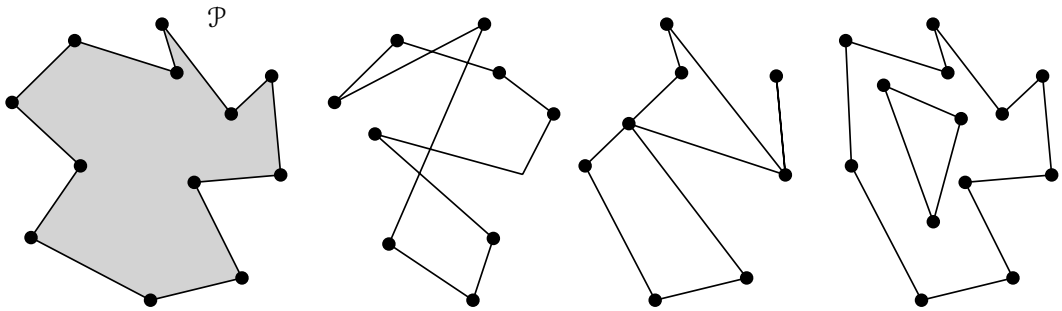

Figure 1.4 – A simple polygon $\mathcal{P}$ to the left, whose interior is shown in gray. The other three objects are not simple polygons.

It can be proven, using the Jordan curve Theorem, that a simple polygon $\mathcal{P}$ divides the plane into two connected regions, the interior and exterior of $\mathcal{P}$. Thus $\mathcal{P}$ is topologically equivalent to a circle, and its interior is topologically equivalent to a disk.

If a simple polygon $\mathcal{P}$ has n vertices, then we will say that $\mathcal{P}$ has size n. Alternatively, we can also say that $\mathcal{P}$ is an n-gon. From now on, unless we say it otherwise, we will *always* assume that a polygon is simple.

Definition 1.2 (Convex polygon). A convex polygon is a simple polygon whose interior is a convex set.

Definition 1.3 (Convex hull). Let P be a non-empty set of points on the plane. The convex hull of P, denoted by $\mathcal{CH}(P)$, is the *smallest* convex polygon containing P.

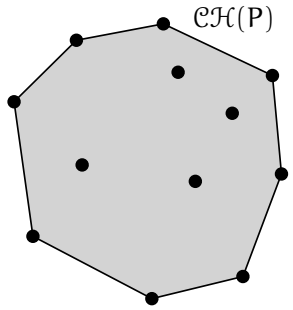

Figure 1.5 – The convex hull $\mathcal{CH}(P)$ of P.

It is easy to prove that the vertices of $\mathcal{CH}(P)$ are elements of P. See Figure 1.5. We will alternatively use *extreme points of* P to refer to vertices of $\mathcal{CH}(P)$. All other points of P will be called *non-extreme* or *interior*.

Definition 1.4 (Plane graph). Given a set of points P, a plane graph G of P is a geometric construction on the plane using the points of P, called the vertices of G, and straight-line segments joining pairs of vertices of G, called the edges of G, such that the following properties hold:

- No vertex of G lies in the strict interior of an edge of G.
- For every pair of vertices of G there is *at most* one edge of G connecting them.
- The vertices of G are the only intersection points among the edges of G.

To match terminology in the literature, we will alternatively say that a plane graph with vertex set P is also a crossing-free structure of P, or defined on P.

Definition 1.5 (Face of a plane graph). Given a plane graph G with vertex set P, we define the faces of G to be the connected components of the complement of G. The unbounded connected component will be called the outer, or unbounded face of G. All other connected components will be called the inner, or bounded faces of G. See Figure 1.6.

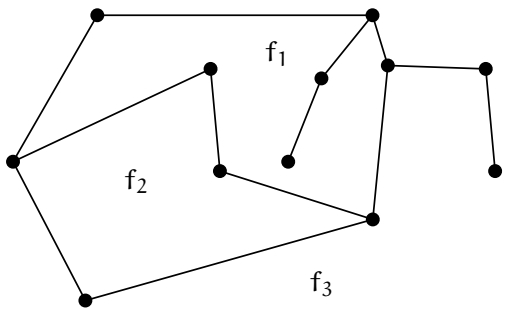

Figure 1.6 – A plane graph with three faces. Face f_3 is the unbounded face.

Given a set P of $n \geqslant 3$ points, we will say that P is in *general position* if and only if no straight-line contains more than two points of P. So, from now on, unless otherwise stated, set P will always be in general position. Also, given a crossing-free structure $\mathcal{S}$ of P, we define the degree of vertex $v \in P$ in $\mathcal{S}$ as the number of edges of $\mathcal{S}$ having v as one endpoint. If $\mathcal{S}$ is clear from the context, we will just talk about the degree of v.

The topics of study in this thesis, and thus also the results, can essentially be divided into two areas: Geometry using Steiner points, and counting algorithms. Let us see each one in turn.

1.2 Geometry using Steiner points

Imagine that we have a certain problem that we want to solve on a given set P of n points on the plane. Say for example that the problem at hand is the computation of a minimum spanning tree of P, or MST for short. A spanning tree $T = T(P)$ of P is a geometric construction that connects all points of P by straight-line segments, called the edges of T, and such that between any pair of points $p, q \in P$, there is *exactly* one path that follows the edges of T. An example of an spanning tree can be seen in Figure 1.7.

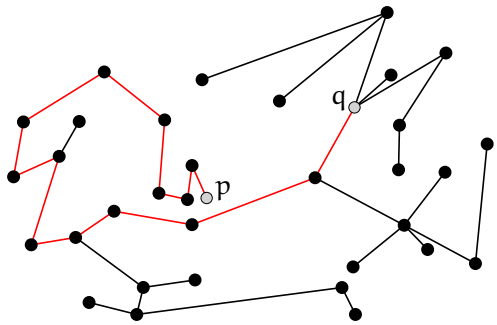

Figure 1.7 – P is the set of black points. A spanning tree T of P is shown with black lines. Observe that for any $p, q \in P$, there is exactly one path between them that follows the edges of T. In addition, tree T as shown is a plane graph.

It is easy to observe that for *any* given non-empty set of points P on the plane, a spanning tree can *always* be constructed, just take one point $p \in P$ and join it to *all* other points of $P \setminus \{p\}$ by straight-line segments. So P by itself is a set of points, but along a spanning tree, set P becomes a set of vertices. In reality, having edges present in a geometric construction we can use the term "point" and "vertex" interchangeably.

If P is given on a metric space, then we can talk about a *minimum* spanning tree of P, that is, a spanning tree where the sum of the lengths of its edges is minimized under the given metric. This sum will be called the "weight" of the tree.

There are, by now, many good algorithms[V] that can compute an MST of P, having set an underlying metric. This is only one of the traditional problems in algorithmic geometry. Now, observe that the way a spanning tree is defined is very natural, yet somewhat restrictive. That is, the edges of the tree must connect pairs of points of P. We could add a little bit more of freedom to this definition and say that the edges of the tree do not necessarily have to connect two elements of P, but they can actually connect any pair of points on the plane, as long as at the end we obtain a tree having

[V] The reader should keep in mind that not *all* algorithms are created equally. Algorithms are no different than many other things in life, there are some better than others.

P as a subset of vertices, *i.e.*, if we denote this tree by $T^* = T_S(P)$, and its set of vertices by $V^* = V(T^*)$, then $P \subseteq V^*$ and $S = V^* \setminus P$. This set S of *extra* points used to construct T^* is known in the literature as a set of *Steiner points*; named after the Swiss mathematician Jakob Steiner (18 March 1796 – 1 April 1863), and T^* is known as a Steiner tree, where P is always clear from context. To the best of our knowledge, the problem that introduced this terminology is precisely the minimum Steiner tree problem[VI], which as the reader can imagine, ask for the Steiner tree of minimum weight of a given set of points P in a metric space. It is not hard to verify that in general, given P, the minimum spanning tree of P, and the minimum Steiner tree are different.

The use of Steiner points has created a paradigm for solving geometric problems. In the case of the minimum Steiner tree, we are interested in improving (optimize) the weight of the a minimum spanning tree of P. There is however another purpose we can use Steiner points for. Imagine that we would like to construct certain geometric structure $\mathcal{S}$ on P that, depending on P, might or might not exist; this is a very common situation in geometry. We could then try to come up with a method to construct a set of Steiner points $S = S(P)$ such that $\mathcal{S}$ can *always* be constructed on $P \cup S$. In such a case, most of the effort is put on making S as small as possible, while keeping $\mathcal{CH}(P)$ intact.

These two ways of using Steiner points have produced many research problems in the area during the past 30 years. As such, these kinds of problems are also responsible for interesting construction and searching techniques that tend to be simple, yet very clever.

The problems studied in Chapters 2 and 3 are the ones related to Steiner points.

1.2.1 Colored quadrangulations with Steiner points – Chapter 2

A quadrangulation of P is a crossing-free structure on P such that the boundary of its outer face coincides with $\mathcal{CH}(P)$, and where *all* bounded faces are empty quadrilaterals, *i.e.*, 4-gons. Now suppose that every point of P is colored with exactly one of $k \geqslant 2$ available colors. We will say that a quadrangulation of P is k-colored if and only if every edge of the quadrangulation joins vertices of different color.

In Chapter 2 we will see that not all k-colored sets of points admit a k-colored quadrangulation. We will show that if P satisfies some condition for the colors of the vertices of its convex hull, then a k-colored quadrangulation can always be constructed on $P \cup S$, where $S = S(P)$ is a k-colored set of Steiner points of size strictly smaller than

[VI] As far as we know, the problem was not posed by Jakob Steiner, and the relation between one and the other is not completely understood.

$\frac{(16k-2)n+7k-2}{39k-6}$. We will also argue that there are k-colored sets of points for which the corresponding k-colored set of Steiner points cannot be smaller than $\frac{n}{3}$.

Our result on the upper bound of S significantly improves on a previous result of S. Kato, R. Mori, and A. Nakamoto. The lower bound is a generalization of the construction shown by this author, T. Sakai, and J. Urrutia in [11] for the particular case when $k = 2$.

A preliminary version of this result was presented at the 28th European Workshop on Computational Geometry (EuroCG 2012). This is a joint work with Prof. Atsuhiro Nakamoto from the Department of Mathematics at Yokohama National University, Japan.

1.2.2 Parity-constrained triangulations with Steiner points – Chapter 3

A triangulation of P is a crossing-free structure on P such that the boundary of its outer face coincides with $\mathcal{CH}(P)$, and where *all* bounded faces are empty triangles. We will say that a triangulation of P is *even* if and only if the degree of *every* vertex is an even number. Similarly, we will say that a triangulation of P is *odd* if and only if the degree of *every* vertex is odd.

In Chapter 3 we will show that there are configurations of points on the plane that admit neither an even triangulation, nor an odd triangulation. However, we will show that we can always construct such triangulations adding at most roughly $\frac{n}{3}$ Steiner points. Moreover, we will show that if we are interested in triangulations where only the interior points of P receive *all* even degree, or *all* odd degree, then we can achieve those construction by using at most roughly $\frac{k}{3}$ Steiner points, where k denotes this time the number of interior points of P.

The problem attacked in Chapter 3 is the Steiner-point version of a problem studied in [4] by O. Aichholzer, T. Hackl, M. Hoffmann, A. Pilz, G. Rote, B. Speckmann and B. Vogtenhuber. There the authors showed, among other results, how to construct triangulations where at least roughly $\frac{2n}{3}$ points of P get even (odd) degree. They also showed that if the assignment of parities to the elements of P is not uniform, *i.e.*, even and odd parities are assigned to the elements of P, then there are configurations where at least roughly $\frac{n}{108}$ parities cannot be satisfied regardless of the chosen triangulation. Due to the interesting applications of even (odd) triangulations, as we will see, it is then attractive to study the Steiner-point version.

A preliminary version of this work was presented at the 26th European Workshop on Computational Geometry (EuroCG 2010).

After Chapters 2 and 3 we will shift to a completely different topic within algorithmic geometry.

1.3 Counting algorithms

So far we have discussed problems in which a crossing-free structure with certain properties is to be constructed on a given set of points P. A natural question that can then be made is: *How many of those crossing-free structures can be found on* P*?* Here we are interested in the *exact* number produced for P. For example, a classical counting problem is that of counting the triangulations of a convex polygon, which, to the best of our knowledge goes back to the Swiss mathematician Leonhard Euler (15 April 1707 – 18 September 1783). Euler discovered a method to compute such number of triangulations but he did not have a formal proof for it. It was many years later, in the late 1830's, that a closed-form solution was found by Eugène Charles Catalan (30 May 1814 – 14 February 1894), a French-Belgian mathematician[VII].

Nowadays the number of triangulations of a convex polygon with $n+2$ sides is known as the n-th Catalan number, usually denoted by C_n, and which can be given directly in terms of binomial coefficients by $C_n = \frac{1}{n+1}\binom{2n}{n}$. The sequence of Catalan numbers is without doubt one of the most popular sequences of natural numbers in combinatorics. In the book Enumerative Combinatorics: Volume 2 by Richard P. Stanley, see [79], one can find a list of 66 problems whose solutions are also Catalan numbers. This list of problems has been continuously updated, and by now, a total of 201 problems related to Catalan numbers are known. This list can be obtained directly from the website of Richard P. Stanley.

For a counting problem nothing is more elegant than a closed-form solution, namely, a formula. Nevertheless, for many counting problems a formula has turned out to be hard to come by. Let us continue with our running example of counting triangulations. If P is in convex position we have a formula for its number of triangulations, but if P has interior points, then things get hard very quickly, to the point that to this day no efficient method is known that can compute the number of triangulations for arbitrary P with interior points, let alone the idea of a closed-form solution. We will soon discuss what we mean by "efficient".

Thus, our main goal now is to focus on developing techniques that correctly compute the number of certain geometric structures. Moreover, we want to perform the counting as fast as possible. Therefore, it is time for us to say something about running times and "efficiency". We mentioned before that no "efficient" method is known that can compute

[VII] Other solutions were found at the same time, see [64].

the number of triangulations for arbitrary P having interior points. We call a counting algorithm "efficient" if its running time can be expressed as a polynomial in n, the size of the input set P.

Now, let us denote by $\mathcal{F}_T(P)$ the class of *all* triangulations of P. Moving away from "efficiency", it is not even known whether triangulations can always be counted considerably faster than enumerating them, *i.e.*, in time $o(|\mathcal{F}_T(P)|)$. There are, however, counting algorithms that experimentally indicate that this is possible, see [2, 70]. Unfortunately a good formal analysis of those algorithms has been very hard to obtain.

In this thesis we are interested in counting not only triangulations, but also other kinds of crossing-free structures, which so far have also been quite difficult to count efficiently.

The topics and results concerning counting algorithms contained in this thesis are the following:

1.3.1 A sweep line algorithm for counting triangulations and pseudo-triangulations – Chapter 4

Let P be again a given set of n points on the plane. While triangulations require by now no introduction, pseudo-triangulations do.

A pseudo-triangle is a simple polygon having *exactly* three convex vertices, that is, the internal angle at those vertices is strictly less than π. An example can be seen to the left in Figure 1.8. A pseudo-triangulation of P is a crossing-free structure on P such that the boundary of its outer face coincides with $\mathcal{CH}(P)$, and where *all* bounded faces are pseudo-triangles. A pseudo-triangulation can be seen to the right in Figure 1.8. Let us denote by $\mathcal{F}_{PT}(P)$ the class of *all* pseudo-triangulations of P.

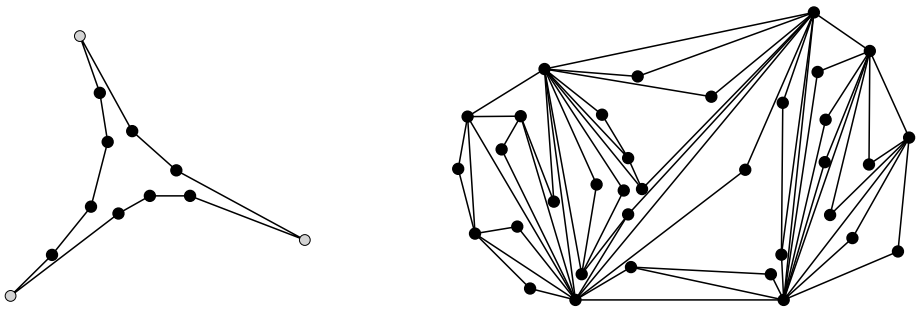

Figure 1.8 – A pseudo-triangle to the left. The three gray vertices are the three convex vertices. A pseudo-triangulation of P can be seen to the right.

In [2] and [6] an algorithm for counting triangulations and pseudo-triangulations, respectively, is shown. Both algorithms are based on the divide-and-conquer paradigm, and both work by finding sub-structures on triangulations and pseudo-triangulations that allow the problems to be split. These sub-structures are called *triangulation paths* for triangulations, or T-paths for short, and *zig-zag paths* for pseudo-triangulations, or PT-paths for short. The formal definition of both sub-structures will be given in Chapter 4. For now we just want to point out that those two algorithms using T-paths and PT-paths have turned out to be very difficult to analyze, to the point that no good analysis of the running time of those algorithms has been presented so far. The interesting thing about those algorithms, besides their simplicity, is that they experimentally indicate that counting can be done in $o(|\mathcal{F}_T(P)|)$ and $o(|\mathcal{F}_{PT}(P)|)$ respectively.

In this chapter we will show two new algorithms, one to compute the number of triangulations of P, and one to compute the number of pseudo-triangulations of P. They are also based on T-paths and PT-paths respectively, but use the sweep line paradigm and not divide-and-conquer. The important thing about our algorithms is that they admit a good analysis of their running times. We will show that our algorithms run in time $O^*(t(P))$ and $O^*(pt(P))$ respectively, where $t(P)$ and $pt(P)$ is the largest number of T-paths and PT-paths, respectively, that the algorithms encounter during their execution. The O^*-notation is like the O-notation, but it neglects polynomial factors. Moreover, by using fancy techniques from combinatorics we will show that $t(P) = O^*(9^n)$, which is the first non-trivial bound on $t(P)$ to be known.

No algorithm like ours was known before, and what makes them even more interesting is that no configuration, large enough, is known such that $t(P)$ and $pt(P)$ are as large as $|\mathcal{F}_T(P)|$ and $|\mathcal{F}_{PT}(P)|$ respectively. It is actually believed that $t(P) = o(|\mathcal{F}_T(P)|)$ and $pt(P) = o(|\mathcal{F}_{PT}(P)|)$, which is supported by many well-studied configurations. We have however failed to prove that in general.

This is a joint work with Karl Bringmann and Saurabh Ray from Saarbrücken, Germany.

1.3.2 Counting triangulations and other crossing-free structures via onion layers – Chapter 5

A crossing-free matching of P is a crossing-free structure on P where *every* vertex has degree at most one. A crossing-free spanning cycle of P is a simple polygon of size n whose vertex set is precisely P.

In Chapter 5 we show yet another new algorithm for counting triangulations which is based on the divide-and-conquer paradigm and the onion layers of P.

Definition 1.6 (Onion layers). Let P be a set of n points on the plane and let $\mathcal{CH}(P)$ denote its convex hull. We define the *onion layers* of P as follows: the first onion layer $P^{(1)}$ of P is $\mathcal{CH}(P)$. For $i > 1$, the i-th onion layer $P^{(i)}$ of P is defined inductively as $\mathcal{CH}\left(P \setminus \bigcup_{j=1}^{i-1} P^{(j)}\right)$. By "number of onion layers of P" we mean the number of *non-empty* onion layers of P, see Figure 1.9.

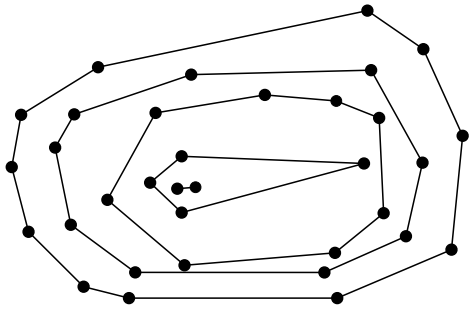

Figure 1.9 – The onion layers of the set of black points are shown with black lines.

It is then easy to observe that for *any* non-degenerate set of n points, the number k of onion layers is *always* at most $\lceil \frac{n}{3} \rceil$.

The algorithm of Chapter 5 for counting triangulations has a running time of the sort $n^{O(k)}$. That is, for configurations having $k = O(1)$ onion layers we obtain a polynomial time algorithm! This is the first algorithm to be known with this property. Moreover, we give an analysis of the algorithm that shows that even when $k = \Theta(n)$, the algorithm has worst-case running time of $O^*(3.1414^n)$. This improves on the worst-case running time of the algorithm of Chapter 4.

We will also show algorithms to count crossing-free matchings, and crossing-free spanning cycles of P in $n^{O(k)}$ time. Unlike the algorithm for counting triangulations of the previous paragraph, this time we are not able to prove a worst-case running time of the sort $O^*(c^n)$, for some positive constant $c \in \mathbb{R}$. The algorithms to count crossing-free matchings and crossing-free spanning cycles retain nonetheless polynomial time whenever $k = O(1)$, which again, was not known before.

The results of Chapter 5 were presented at the 28th Annual ACM Symposium on Computational Geometry (SoCG 2012). This is a joint work with Karl Bringmann, Saurabh Ray, and Radu Curticapean.

1.3.3 Miscellaneous results on counting triangulations – Chapter 6

Towards the end of the thesis, in Chapter 6, we will show some miscellaneous results on counting triangulations. The first result that will be shown is an approximation algorithm for counting triangulations. This algorithm fails to count triangulations exactly, but it provides in sub-exponential time, $2^{o(n)}$, an approximation Λ such that $|\mathcal{F}_T(P)| \leqslant \Lambda \leqslant |\mathcal{F}_T(P)| \cdot 2^{o(n)}$. Since it is known that $|\mathcal{F}_T(P)| = c^n$, for some positive constant $c \in \mathbb{R}$, we have that $c \leqslant \Lambda^{\frac{1}{n}} \leqslant c^{1+o(1)} \leqslant (1 + o(1))c$.

The second result shown in Chapter 6 is a hardness result. Let E be some set of edges on P, that is, every edge $e \in E$ has its endpoints in P. It is known that the problem of deciding whether a triangulation of P can be constructed using only edges of E is NP-complete, see [53, 74]. If we again denote by k the number of onion layers of P, we will prove that this problem is W[2]-hard, when k is considered the parameter of the problem. This means that no algorithm with a running time of the sort $g(k) \cdot n^{O(1)}$ exists for this problem unless FPT $=$ W[2]. The complexity classes FPT and W[2] are well-known complexity classes in the area of parameterized algorithms. Probably the most important open question in that area is whether FPT $=$ W[2]. It is however widely believed that FPT $\neq$ W[2]. The book [37] of J. Flum and M. Grohe is the standard reference in parameterized complexity theory.

Observe that the problem of deciding whether a triangulation of P can be constructed using only edges of E can be trivially reduced to the counting version, where we are interested in counting the triangulations of P that can be formed using only edges of E. So the counting version is also W[2]-hard.

We close Chapter 6, and thus also the thesis, by showing some experiments comparing the algorithms for counting triangulations of Chapters 4 and 5 with the algorithm presented in [70], which is reported to be very fast in practice.

The approximation algorithm is a joint work with Karl Bringmann, Saurabh Ray, and Raimund Seidel.

The hardness result appeared along with the results of Chapter 5 at the 28th Annual ACM Symposium on Computational Geometry (SoCG 2012).

1.4 A quick word on the model of computation

Throughout this thesis we will assume that we are working on the real-RAM model of computation. That is, our algorithms are designed for a machine that can hold in each cell of memory a real number, regardless of how big this number really is. Also, we can

access any cell of memory in unit time, and for a pair of real numbers we can perform common operations in unit time such as the following: $+$, $-$, $*$, $/$, $=$, $\neq$, $<$, $>$, $\geqslant$, $\leqslant$. Also operations like $\sqrt[k]{n}$, $\exp(n)$, $\log(n)$ can be performed in unit time if needed.

With this assumptions, many geometric primitives can be correctly implemented so they can be performed in $O(1)$ time, such as:

- Given three points p, q, r, decide if $p \to q \to r$ does a right turn or a left turn.
- Given a point and a line, decide on which side of the line the point lies.
- Given two straight-line segments, decide if they intersect.

The main idea of working in the real-RAM model is that we abstract from a problem its inherent difficulty, that is, we do not care how those operations are performed, but rather what do we do with them, how do we use them to achieve our goals (constructions). Thus the real-RAM model could make, in any case, implementation of algorithms just harder (more technical), but not impossible. For example, arithmetic on large numbers can be implemented efficiently, see [56]. The polynomial overheads incurred by these operations will be, nevertheless, swallowed by the O^*-notation. We refer the reader to [67] for more information about the real-RAM model, as well as [55, 59] for other technical aspects on algorithm implementation.

CHAPTER 2

COLORED QUADRANGULATIONS WITH STEINER POINTS

Quadrangulations of sets of points received extensive attention back in the 90's, where they were sometimes preferred over triangulations in the study of finite element methods and scattered data interpolation, see [51] for example. It is not hard to see that *not every* given set P of n points admits a quadrangulation. It can however be verified that necessary conditions for P to admit a quadrangulation are (1) $|P| \geqslant 4$, and (2) the size of the convex hull $\mathcal{CH}(P)$ of P must be of even cardinality. It turns out that these two conditions are also sufficient, see [21, 68]. Thus, given *any* set of points P, at most *one* Steiner point s needs to be added to P so that $P \cup \{s\}$ admits a quadrangulation.

Having characterized the set of points that admit quadrangulations, and having designed optimal algorithms for their computation, in $\Theta(n \log n)$ time, researchers started looking for quadrangulations of set of points having special properties, for example, that each face of the quadrangulation must be a convex quadrilateral, see [22, 20, 45, 73] for example. Already in this setting it was shown in [22] that again, not every set of points admits a *convex quadrangulation*, thus the use of Steiner points is again required if one insists on constructing one on the given set of points. Hence the question now is not whether a given set of points P admits a convex quadrangulation, but rather how many Steiner points are sufficient and how many necessary in order to construct one. It was shown in [22] that one can always construct a convex quadrangulation using at most $\frac{3n}{2}$ *interior Steiner points*[1], and that $\frac{n}{4}$ are sometimes necessary. Later, in [45],

[1] These are Steiner points that are introduced in the interior of the convex hull of the given set of points.

both bounds were improved to roughly $\frac{5n}{4}$ and $\frac{n}{3}$ respectively. For experimental studies on convex quadrangulations we refer the reader to [20, 73], and to [82] for a somewhat dated survey on quadrangulations.

Another special kind of quadrangulations arises when the input set of points P is colored using $k \geqslant 2$ colors, and we look for a quadrangulation not containing monochromatic edges, that is, the quadrangulation should be a properly colored plane graph. We will refer to such quadrangulations as k-*colored quadrangulations*, for the special case when $k = 2$ we will alternatively use the term *bichromatic quadrangulation*. At this point is very important to say that since monochromatic edges are forbidden in a k-colored quadrangulation, and we are regarding $\mathcal{CH}(P)$ as being the outer cycle of *any* quadrangulation of P, then we will assume from now on, and to avoid any obvious complication, that $\mathcal{CH}(P)$ is a properly colored polygon.

As in the monochromatic case, one can again come up with configurations not admitting k-colored quadrangulations, thus again requiring the use of Steiner points. The bichromatic configuration to the left in Figure 2.1 is taken from [25].

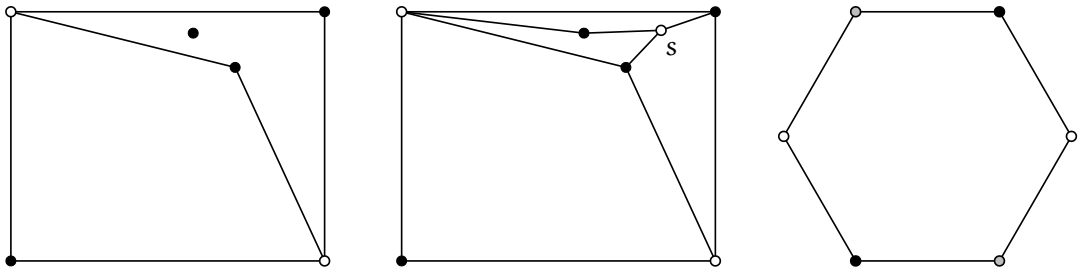

Figure 2.1 – To the left a bichromatic set of points not admitting a bichromatic quadrangulation without the use of Steiner points. In the middle the same configuration quadrangulated with one Steiner point s. To the right a 3-colored configuration not admitting a 3-colored quadrangulation regardless the number of Steiner points used.

The study on k-colored quadrangulations of sets of points is rather new. It was shown in [11] that one can always construct a bichromatic quadrangulation with the use of roughly $\frac{5n}{12}$ interior Steiner points, and that $\frac{n}{3}$ Steiner points are sometimes necessary. They also considered the case $k = 3$ and showed a surprising fact, there are 3-colored sets of points that do not admit 3-colored quadrangulations regardless of the number of interior Steiner points used, which is definitely an unexpected result! The configuration presented in [11] is shown to the right in figure 2.1.

The strange phenomenon of not admitting 3-colored quadrangulations, even with the use of Steiner points, was recently explained in [48], where the authors showed an elegant

characterization of the 3-colored sets of points that admit 3-colored quadrangulations using a *finite* number of interior Steiner points. In the same paper, the authors showed that if possible, a 3-colored quadrangulation can be constructed with the use of at most $\frac{7n+17m-48}{18}$ interior Steiner points, where $|P| = n$ and $|\mathcal{CH}(P)| = m$. Note however that this number depends on the size of $\mathcal{CH}(P)$, and can get large whenever m and n are comparable in size. For example, if $m = \frac{3n}{4}$, then the bound becomes $\frac{79n-192}{72}$ which is larger than n already when $n \geqslant 5$.

2.1 Our contribution

In this chapter we show our contribution to the problem of k-colored quadrangulations using Steiner points, with $k \geqslant 3$. We will show how to use the techniques of [11] for the bichromatic case to obtain a simple algorithm for the k-colored case. Our algorithm uses fewer than $\frac{(16k-2)n+7k-2}{39k-6}$ interior Steiner points to construct a k-colored quadrangulation of a given k-colored set of points P, provided that $\mathcal{CH}(P)$ is properly colored. Our bound on the number of used Steiner points has the following advantages:

(1) Our algorithm improves on the algorithm shown in [48], since, as we will see, it performs equally well when $\mathcal{CH}(P)$ is small, but it improves the worst-case behavior when $\mathcal{CH}(P)$ is large. For comparison, our bound for $k = 3$, at worst, is essentially $\frac{46n}{111} < \frac{5n}{12}$, while the one presented in [48] can grow larger than n if the right conditions are met.

(2) Our bound represents the first bounds for the cases when $k \geqslant 4$.

We will divide the chapter as follows: In § 2.2 we give the necessary definitions and the precise statement of our result. In § 2.3 we prove our main theorem, and in § 2.4 we briefly discuss the extension of the lower bound of $\frac{n}{3}$ interior Steiner points of [11] for the general case $k \geqslant 3$.

2.2 Preliminaries

In order to make the presentation more self-contained, we will state the results from other papers that will be used, and will be referred to. Let us first start with some terminology.

Let $Q \subset \mathbb{R}^2$ be an m-sided simple polygon, with $m \geqslant 4$ even, and suppose that Q is properly k'-colored, where $k' \geqslant 2$. Let us assume that the k' chromatic classes used to color the vertices of Q are $1, 2, \ldots, k'$, and that they all appear in Q. Let us denote the

color of a vertex v of Q by $c(v)$. Let us define an orientation $\mathcal{O}$ for the edges of Q as follows: If $e = uv$ is an edge of Q, then we orient e from u to v if $c(u) < c(v)$, and from v to u otherwise. Let $e^+_{\mathcal{O}}(Q)$ and $e^-_{\mathcal{O}}(Q)$ be the number of edges of Q in clockwise and counter-clockwise direction, respectively, w.r.t. orientation $\mathcal{O}$.

Definition 2.1 (Winding number). Let Q and $\mathcal{O}$ be as explained above. The winding number of Q, denoted by $\omega(Q)$, is defined as:

$$\omega(Q) = |e^+_{\mathcal{O}}(Q) - e^-_{\mathcal{O}}(Q)|$$

for $k' = 3$, and $\omega(Q) = 0$ for $k' \neq 3$.

Observe that the winding number of a polygon Q is non-trivial only when Q is 3-colored.

For a k-colored set of points P, $k \geqslant 2$, we will use $\omega(P)$ as a shorthand for $\omega(\mathcal{CH}(P))$, extending the definition of winding number for polygons to sets of points. Observe however that $\omega(\mathcal{CH}(P))$ depends solely on the number $k' \leqslant k$ of colors appearing on $\mathcal{CH}(P)$, so we could have $\omega(P) = \omega(\mathcal{CH}(P)) \neq 0$ while $k \neq 3$. Finally, we will say that P can be k-*quadrangulated* if P admits a k-colored quadrangulation.

The following result is the one, mentioned before, that characterizes the 3-colored sets of points which can be 3-quadrangulated with Steiner points added [48].

Theorem 2.1 (S.Kato, R. Mori, A. Nakamoto). *Let $P \subset \mathbb{R}^2$ be a 3-colored set of n points in general position such that $|\mathcal{CH}(P)| = m$. Then there exists a set $S = S(P)$ of Steiner points such that $P \cup S$ can be 3-quadrangulated if and only if $\omega(P) = 0$. In such a case we have that $|S| \leqslant \frac{7n+17m-48}{18}$.*

Now we can easily decide whether a 3-colored set of point admits a 3-colored quadrangulation. Nevertheless, as we mentioned before, the number of Steiner points required by Theorem 2.1 can get larger than n when m and n are comparable in size.

Our main contribution is the following result:

Theorem 2.2 (V. Alvarez, A. Nakamoto). *Let $P \subset \mathbb{R}^2$ be a k-colored set of n points in general position, where $k \geqslant 2$. If $\omega(P) = 0$ or $k \geqslant 4$, then there exists a set $S = S(P)$ of Steiner points such that $P \cup S$ can be k-quadrangulated, and $|S| < \frac{(16k-2)n+7k-2}{39k-6}$.*

The condition $\omega(P) = 0$ or $k \geqslant 4$ in the previous statement means, as we will see, that even when only three colors appear on $\mathcal{CH}(P)$, and they cause $\omega(P) \neq 0$, we can still find a set $S = S(P)$ of Steiner points such that $P \cup S$ can be k-quadrangulated as long as we have at least four colors in total at our disposal.

Note that our result besides of being able to work with more than three chromatic classes, depends only on n and k, which is a great improvement over the previously known bound for $k = 3$.

2.3 Proof of Theorem 2.2

In order to prove our theorem we will need some intermediate results, the first one is easily proven using the well known Euler's formula:

Lemma 2.1. *Let* $P \subset \mathbb{R}^2$ *be a set of* n *points such that* $|\mathcal{CH}(P)| = m$. *Then any quadrangulation of* P *has* $(n-1) - \frac{m}{2}$ *quadrilaterals and* $2(n-2) - \frac{m}{2}$ *edges.*

The following lemma was shown in [48]:

Lemma 2.2. *Let* $Q \subset \mathbb{R}^2$ *be a 3-colored simple polygon colored by colors* c_1, c_2, c_3. *Then the winding number of* Q *is invariant for any bijection from* $\{c_1, c_2, c_3\}$ *to* $\{1,2,3\}$.

That is, the winding number is well-defined, and we may well assume that if Q is a 3-colored simple polygon, then it is colored by $\{1,2,3\}$. We now have the following lemma, which, as we will see, is the one that will make everything go through:

Lemma 2.3. *Let* $Q \subset \mathbb{R}^2$ *be a properly* k*-colored simple polygon of* $m \geqslant 4$ *sides such that* $\omega(Q) = 0$. *Then* Q *can be partitioned into* $r = \frac{m-2}{2}$ *properly colored quadrilaterals* $Q_1, \ldots, Q_r$ *such that* $\omega(Q_i) = 0$ *for every* $1 \leqslant i \leqslant r$.

Proof. We will proceed by induction over m. The case $m = 4$ can easily be verified, thus we will directly assume that Lemma 2.3 holds for every $m' < m$, and we will prove it when $m' = m$.

We will divide the proof into two parts, the first one being when Q is exactly 3-colored, and the second one when Q is at least 4-colored.

(1) By Lemma 2.2, we may assume that the chromatic classes are exactly $\{1,2,3\}$. Observe that there is a vertex $v \in Q$ such that its two neighbors are of the same color. For otherwise, *i.e.*, if every vertex of Q has two neighbors with distinct colors, then we can easily check that Q has a periodic cyclic sequence of colors 1,2,3, which is contrary to $\omega(Q) = 0$. See to the left in Figure 2.2.

Now assume that all edges of Q are oriented using the orientation $\mathcal{O}$ explained before. Let $v \in Q$ be a vertex with two neighbors $u, w \in Q$ of the same color, where u is the right neighbor of v, and w the left neighbor. Let $x \in Q$ be the right neighbor of u. Since Q is properly colored, x has a color distinct from those of u and w, and hence we can add an edge wx to create the properly colored quadrilateral $Q_1 = xuvw$. Now, let Q' be the convex polygon defined by $Q \setminus \{u, v\}$. We first observe that $\omega(Q_1) = 0$ since u and w have the same color. Second, note

Figure 2.2 – If Q is colored by the cyclic sequence 1,2,3, as shown to the left, it can be easily verified that $\omega(Q) \neq 0$.

that $\omega(Q') = 0$ as well, which can be explained as follows: Since u and w have the same color, the orientations of the two edges vw and vu are canceled in the computation of $\omega(Q)$. Moreover, the edges xu and xw are both oriented away from x, so xw is the actual edge making $\omega(Q') = 0$. Hence we get $\omega(Q) = \omega(Q') = 0$.

We can now repeat these procedures inductively on Q', as shown to the right in Figure 2.2.

(2) Now let us assume that Q is at least 4-colored. We now claim that there is at least one vertex $w \in Q$ such that at least one of its neighbors at distance 3 on Q, in clockwise or counter-clockwise order, is of different color. Indeed, assume otherwise and note that if every vertex of Q shares the same color with its two neighbors at distance 3 on Q, then Q would be 3-colored and thus we would not be having this conversation. See to the left in Figure 2.2.

Let $w \in Q$ be one of the vertices having a neighbor at distance 3 of different color, say such a neighbor is $x \in Q$. Note that we can join w and x with a straight line, thus creating the quadrilateral $Q_1 = wvux$, where $v, u \in Q$ are the vertices at distance 1 and 2 from w respectively. Let Q' be defined as in (1). If $\omega(Q') = 0$ we are done again. Otherwise, if $\omega(Q') \neq 0$, then Q' must be 3-colored and the fourth color appears at either v or u. We can thus shift (rotate) the labels of the vertices of Q at most two positions, either clockwise or counter-clockwise, so that the fourth color appears at vertex w after the shift. At his point Q' is 4-colored and we would be done again. See Figure 2.3.

In both cases that the total number of created quadrilaterals is $\frac{m-2}{2}$ follows from Lemma 2.1. ■

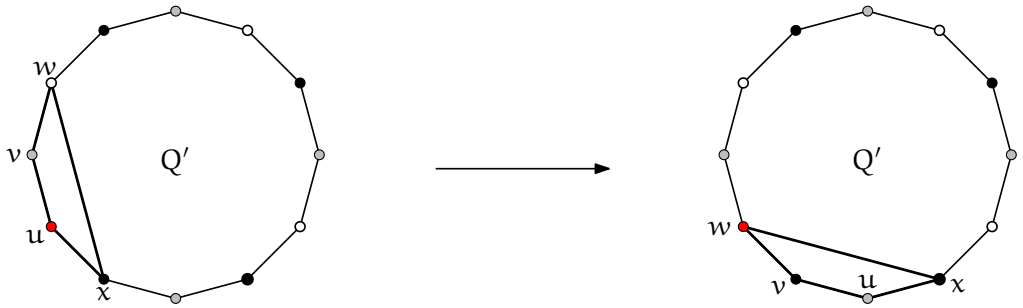

Figure 2.3 – Rotation of labels counter-clockwise so that the fourth color of Q appears again on Q′.

The last result we need from [48] is the following:

Lemma 2.4. *Let* $P = c_1 \cup c_2$ *be a 2-colored set of* n *points on the plane such that* $|\mathcal{CH}(P)| = m$, *where* c_1 *and* c_2 *are the color classes of* P *such that* $|c_1| \geqslant |c_2|$. *Then there exists a set* $S = S(P)$ *of Steiner points such that* $P \cup S$ *can be 2-quadrangulated, and* $|S| \leqslant \left\lfloor \frac{|c_1|}{3} \right\rfloor + \left\lfloor \frac{|c_2|-(m/2)}{2} \right\rfloor \leqslant \frac{5n}{12} - 1$.

The previous Lemma is essentially one of the main results of [11], and it is proven using exactly the same techniques as for Theorem 1 of [11], however, they are applied differently so the constant term on the bound of $|S|$ is improved in the worst case from $(-1/3)$, in [11], to -1, in [48]. This negligible improvement of constants will play a useful role when proving Theorem 2.2.

The next lemma is the last one before we proceed with the proof of our main theorem.

Lemma 2.5. *Let* $P \subset \mathbb{R}^2$ *be a* k*-colored set of* $(q+4)$ *points such that* $|\mathcal{CH}(P)| = 4$ *and* $k \geqslant 2$. *Then there exist two sets of Steiner points* $S_\Gamma = S_\Gamma(P)$ *and* $S_\Delta = S_\Delta(P)$ *such that:*

- $P \cup S_\Gamma$ *can be* k*-quadrangulated, and* $|S_\Gamma| \leqslant \frac{5q+8}{12}$.
- $P \cup S_\Delta$ *can be* k*-quadrangulated, and* $|S_\Delta| < \frac{(2k+1)q+16k}{6k}$.

Proof. Let us divide the proof into two parts, one considering S_Γ and the other considering S_Δ. For simplicity, let us denote $\mathcal{CH}(P)$ by Q.

- Note that P can be regarded as a bichromatic set of points as follows: If Q is bichromatic itself, say using colors c_1, c_2, then we can recolor every interior point of color different than c_2 with color c_1. We will rename the chromatic classes as $c_\alpha = c_1$ and $c_\beta = c_2$.

If Q is 3-colored, say using colors c_1, c_2, c_3, then one color must appear twice on Q, say without loss of generality c_2. Proceed as before, recolor every point of color different than c_2 with a new color c_α. Rename the chromatic class c_2 as c_β.

If Q is 4-colored, say using colors c_1, c_2, c_3, c_4, assume that c_1, c_3 and c_2, c_4 appear in diagonally opposite vertices of Q in clockwise order. Now recolor P with two new colors c_α and c_β as follows: Every point of color c_2, c_4 receives color c_β. The rest of the points receive color c_α.

As we end up having a bichromatic set of points, using colors c_α, c_β, say without loss of generality that $|c_\beta| \leqslant |c_\alpha|$. Thus by Lemma 2.4 there exists a set $S_\Gamma = S_\Gamma(P)$ of Steiner points such that $P \cup S_\Gamma$ can be 2-quadrangulated and:

$$|S_\Gamma| \leqslant \left\lfloor \frac{|c_\alpha|}{3} \right\rfloor + \left\lfloor \frac{|c_\beta| - 2}{2} \right\rfloor \leqslant \frac{5|P|}{12} - 1 = \frac{5q + 8}{12}$$

- Let us now do the following: Say without loss of generality that c_1 is the smallest chromatic class among the k chromatic classes. Let us assume that Q is colored with colors other than c_1, we will see later on that this assumption only worsens the upper bound. Now let us introduce two Steiner points of color c_1 inside Q, very close to two opposite vertices of Q, and in such a way that we create a new quadrilateral Q' that is still properly colored and still contains the q interior points. Let P' be the set of points formed by the vertices of Q' and the q points in its interior, see Figure 2.4.

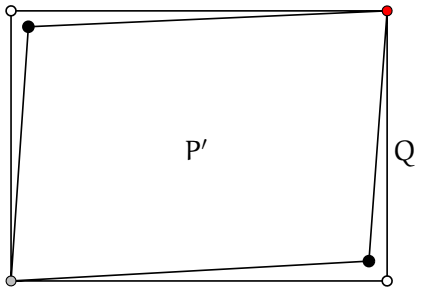

Figure 2.4 – Points colored with color c_1 are represented in black. Quadrilateral Q' still contains the q interior points that quadrilateral Q originally contained.

Now recolor every point of P' of color different than c_1 with a new color c. This leaves only two chromatic classes, c_1 and c, where c_1 is still the smallest one. We can now proceed with the quadrangulation of a bichromatic point set again, this

time finding a set of Steiner points $S_\Delta = S_\Delta(P)$ such that:

$$\begin{aligned}|S_\Delta| &\leqslant \left\lfloor \frac{|c|}{3} \right\rfloor + \left\lfloor \frac{(|c_1|+2)-2}{2} \right\rfloor + 2 \\ &\leqslant \frac{|c|}{3} + \frac{|c_1|}{2} + 2 = \frac{|c|+|c_1|}{3} + \frac{|c_1|}{6} + 2 \\ &< \frac{q+2}{3} + \frac{q}{6k} + 2 \\ &= \frac{q(2k+1)+16k}{6k}\end{aligned}$$

where the first inequality is obtained using Lemma 2.4 again. The last inequality is obtained by $|c| = q+2-|c_1|$ and the assumption that c_1 is the smallest chromatic class, so $|c_1| < \frac{q}{k}$. If $|c_1| = \frac{q}{k}$, then we have $|c_1| = \cdots = |c_k| = \frac{q}{k}$, and hence we can take c_1 so that c_1 does appear on Q. In this case, only at most one Steiner point is required in the beginning to obtain Q'. Hence we would obtain $|\mathbb{S}_\Delta| \leqslant \frac{q(2k+1)+7k}{6k}$ which is slightly smaller, but it would still play a role reducing the bound on Theorem 2.2. ■

We are finally ready to prove Theorem 2.2:

Proof. Let P be a k-colored set of n points where $|\mathcal{CH}(P)| = m$ and $q = n - m$ are interior points. If $\omega(P) = 0$, by Lemma 2.3 we know that we can partition $\mathcal{CH}(P)$ into $r = \frac{m-2}{2}$ convex quadrilaterals Q_i, $1 \leqslant i \leqslant r$, each of which is properly colored and has $\omega(Q_i) = 0$. If $\omega(P) \neq 0$, then by Theorem 2.1 the only case that makes sense is $k \geqslant 4$. That is, P is colored with at least four colors but only three of them appear in $\mathcal{CH}(P)$, causing $\omega(P) \neq 0$. In this case we cannot apply Lemma 2.3 directly, so we will introduce one Steiner point s inside $\mathcal{CH}(P)$, very close to one vertex v of $\mathcal{CH}(P)$ such that s replaces v in $\mathcal{CH}(P)$. If the color of s is chosen such that the new $\mathcal{CH}(P)$ is 4-colored, and observe that this is always the case, we can proceed with Lemma 2.3 as before.

Let q_i be the number of interior points in quadrilateral Q_i. By Lemma 2.5 we know that there are two ways of k-quadrangulating Q_i using Steiner points. The first way uses a set of Steiner points S_Γ^i for each Q_i, which would overall give a k-quadrangulation of $P \cup S_\Gamma$ with a set $S_\Gamma = S_\Gamma^1 \cup \cdots \cup S_\Gamma^r$ of Steiner points such that:

$$|S_\Gamma| = \sum_{i=1}^{r} |S_\Gamma^i| \leqslant \sum_{i=1}^{r} \frac{5q_i+8}{12} = \frac{2r}{3} + \sum_{i=1}^{r} \frac{5q_i}{12} = \frac{m-2}{3} + \frac{5q}{12} \tag{2.1}$$

The second way of k-quadrangulating Q_i using Steiner points would give a k-quadrangulation of $P \cup S_\Delta$ using the set of Steiner points $S_\Delta = S^1_\Delta \cup \cdots \cup S^r_\Delta$ such that:

$$\begin{aligned} |S_\Delta| &= \sum_{i=1}^{r} |S^i_\Delta| < \sum_{i=1}^{r} \frac{(2k+1)q_i + 16k}{6k} = \frac{8r}{3} + \sum_{i=1}^{r} \frac{(2k+1)q_i}{6k} \\ &= \frac{4(m-2)}{3} + \frac{(2k+1)q}{6k} \end{aligned} \tag{2.2}$$

There is something important to note here. We are assuming that in each Q_i all the chromatic classes appear. If that is not the case, say there is at least one chromatic class not appearing in some Q_j, $1 \leqslant j \leqslant r$, then the size of the smallest chromatic class in Q_j is 0. In such a case, as the reader can verify, we would obtain an improvement on $|S^j_\Delta|$, which would in turn improve $|S_\Delta|$.

We would like to see now which set of Steiner points, among S_Γ and S_Δ, performs better, and under what circumstances. For the following we remind the reader that $q = n - m$. If $|S_\Gamma| \leqslant |S_\Delta|$ we have:

$$\frac{m-2}{3} + \frac{5(n-m)}{12} < \frac{4(m-2)}{3} + \frac{(2k+1)(n-m)}{6k}$$

and hence $m > \frac{k(n+24)-2n}{13k-2}$. The bound on m in turn implies $q < \frac{12k(n-2)}{13k-2}$.

Let S be a set of Steiner points added to P so that $P \cup S$ admits a k-quadrangulation. Estimate $|S|$ by $\min\{|S_\Gamma|, |S_\Delta|\}$. Then, if $m > \frac{k(n+24)-2n}{13k-2}$ we obtain:

$$|S| \leqslant |S_\Gamma| + 1 = \frac{m-2}{3} + \frac{5q}{12} + 1 = \frac{4n+q+4}{12} < \frac{(16k-2)n + 7k - 2}{39k-6}$$

where the second equality follows by substituting $m = n - q$. On the other hand, if $m \leqslant \frac{k(n+24)-2n}{13k-2}$, then:

$$\begin{aligned} |S| \leqslant |S_\Delta| + 1 &< \frac{4(m-2)}{3} + \frac{(2k+1)q}{6k} + 1 = \frac{(6k-1)m + (2k+1)n - 10k}{6k} \\ &< \frac{(16k-2)n + 7k - 2}{39k-6} \end{aligned}$$

where the equality follows by substituting $q = n - m$.

As a final remark note that if by any chance $\mathcal{CH}(P)$ is small, say of constant size, then the bound given by equation (2.2) is better than the one given by equation (2.1) for $k \geqslant 3$.

Theorem 2.2 now follows entirely. ■

2.4 Closing remarks and conclusions

It was shown in [11] that there are bicolored sets of $n = 3m$ points, with $m \geqslant 4$, that require at least m Steiner points to be 2-quadrangulated, where $|\mathcal{CH}(P)| = m$. See in the left upper corner of Figure 2.5 for a reference of a configuration like that.

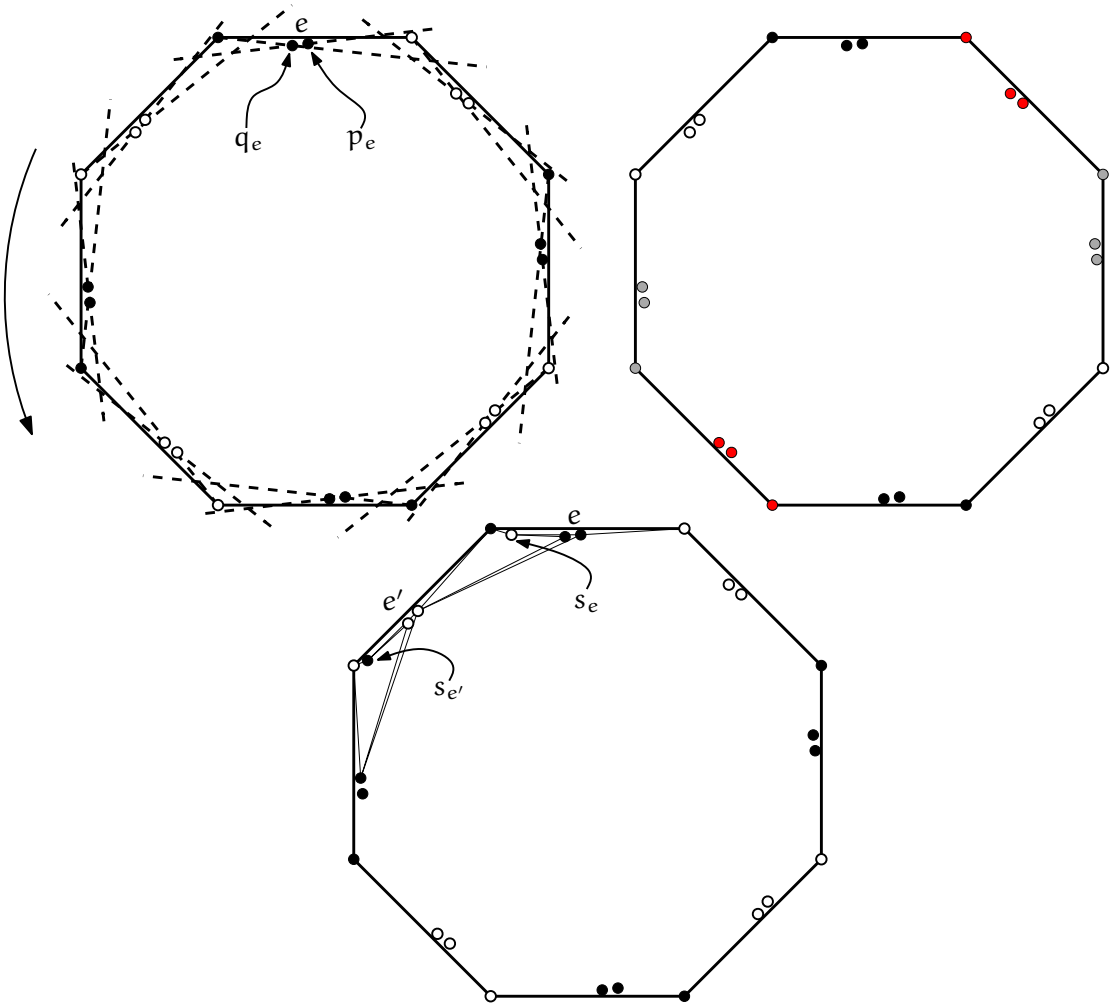

Figure 2.5 – In the left upper corner the bichromatic configuration P that needs at least $\frac{n}{3}$ Steiner points in order to be 2-quadrangulated. Every edge e of $\mathcal{CH}(P)$ gets associated with a pair of interior points p_e, q_e. Down in the middle a partial bichromatic quadrangulation using Steiner points $s_e \neq s_{e'}$ is shown. In the right upper corner the same configuration colored with 4 colors.

We can briefly describe the configuration, as presented in [11], as follows: Each edge e of $\mathcal{CH}(P)$ gets associated with exactly two interior points p_e, q_e, what pair of interior

points gets associated to e is also shown in the left upper corner of Figure 2.5. The coloring of the configuration is done in the following way: The endpoints of e get different colors and its associated pair of interior points get the color of the left endpoint of e, as seen by the reader.

The idea behind the proof of [11] is essentially to check that for every edge e, there is one Steiner point s_e that can be charge to the triple of points formed by p_e, q_e and the left point of e, observe that this triple of points is monochromatic. Intuitively, the Steiner point s_e is the one that locally helps to k-quadrangulate the region between e and p_e, q_e, see down in the middle in Figure 2.5. The proof of [11] is done by cases. For the k-colored case the number of cases to analyze increases, since now there are more colors to choose from for the Steiner points s_e. Nonetheless, the interested reader will be able to verify that the same arguments of [11] carry over into the k-colored setting, and hence also the lower bound of $\frac{n}{3}$ Steiner points. We will thus refrain ourselves from repeating those arguments here. See in the right upper corner of Figure 2.5 for an example of what a 4-colored configuration could look like.

2.4.1 Conclusions

In this chapter we studied the problem of constructing k-colored quadrangulations of k-colored set of points using Steiner points, with $k \geqslant 3$. We were able to improve the previous known upper bound on the number of Steiner points when $k = 3$ with a general method that also provides the first upper bounds for the case when $k \geqslant 4$. We also pointed out that the lower bound of $\frac{n}{3}$ interior Steiner points of [11] for the bichromatic case follows quasi-verbatim to the more general case $k \geqslant 3$.

The upper bound on the number of Steiner points we presented for the k-colored case is always less than $\frac{(16k-2)n+7k-2}{39k-6}$, which, ignoring lower degree terms, is essentially $\frac{16n}{39} \approx 0.4102n$. Since $\frac{n}{3}$ is the lower bound on the same number, both bounds, upper and lower, are off by roughly $\frac{n}{13}$, which albeit being small, is not desirable. Therefore closing this gap is still an interesting open question, we believe that the correct bound should be on the lower end.

There is one more thing to note. The upper bound of [11] for the bichromatic case is roughly $\frac{5n}{12} = 0.41\bar{6}n$. Thus both upper bounds, ours and of [11], are essentially the same in the worst case. This is because our algorithm has at its core the algorithm for the bichromatic case. Hence, an improvement of bounds for the bichromatic case will carry over into the general case using our algorithm, as long as only interior Steiner points are used. We believe that the cases $k = 2$ and $k = 3$ are really challenging, while the cases $k \geqslant 4$ might be more attackable.

CHAPTER 3

PARITY-CONSTRAINED TRIANGULATIONS WITH STEINER POINTS

Let $P \subset \mathbb{R}^2$ be a set of n points in general position and let $\Gamma : P \to \{0,1\}$ be an assignment of parities to the elements of P, where 0 means even and 1 means odd. Let G be a straight-edge plane graph with vertex set P. We say that a vertex $v \in P$ of G is *happy*, with respect to G, if and only if the degree of v in G is of the parity assigned to v by Γ. If a vertex is not happy w.r.t. G we will say that v is *unhappy*. If the graph G is clear from context we will just say that vertices are happy or unhappy without referring to G.

Given P and Γ, the problem of finding plane graphs on P that maximize the number of happy vertices has recently received some attention. In [4] it was shown that one can always construct a tree, a 2-connected outerplanar graph, and a pointed pseudo-triangulation that makes all but at most three vertices happy. For the class of triangulations it was shown that one can always construct one that makes essentially $\frac{2n}{3}$ of its vertices happy, but a configuration of points with parities was also shown where at least $\frac{n}{108}$ vertices remain unhappy regardless of the chosen triangulation. The construction of this lower bound requires the use of both parities, but the authors pointed out that there are two particular cases that might accept triangulations with as many as $n - o(n)$ happy vertices. These two particular cases are the ones where the parities assigned to the elements of P by Γ are either *all* even or *all* odd respectively. However, showing that in those particular cases $n - o(n)$ vertices can be made happy turned out to be very challenging. In the same paper the authors showed that in the *all-even* case, a tri-

angulation that makes at least $\frac{2n}{3}$ vertices happy can always be constructed. They also showed that in the *all-odd* case a $\frac{10}{13}$ fraction of happy vertices can always be ensured.

These two special cases, all even or all odd, are of significant interest since they have interesting properties and/or applications. For example, it is well known that a connected graph G having all its vertices of even degree is Eulerian. If on top of it G happens to be a triangulation as well, then G is also 3-colorable, see [44, 80] for a general reference on 3-colorable planar graphs. Those two properties are usually considered "nice" in a graph, and they are characterized only by the parity of the degree of its vertices. For 3-colorability of triangulations a slightly weaker characterization is known: T is a triangulation having all its interior vertices of even degree if and only if T is 3-colorable, see [30] for example. The application related to the all-odd case is a little bit more intricate and we refer the reader to [5] where this application is shown.

Let P be as before. We will say that a triangulation T of P is *even*, or *odd*, if and only if the degree of *every* vertex of T is even, or odd respectively. If only the interior vertices of T are even, or odd, we will say that T is *pseudo-even*, or *pseudo-odd* respectively. This defines four kinds of triangulations of P: Even, pseudo-even, odd, pseudo-odd.

In this chapter we attack the following problem: Given P and one kind $\mathcal{T}$ of triangulations of the four mentioned above, construct a set $S = S(P, \mathcal{T})$ such that a triangulation of kind $\mathcal{T}$ can *always* be constructed on $P \cup S$.

Thus, the problem attacked in this chapter can be seen as the Steiner-point version of the ones regarding triangulations presented in [4].

With the idea of using Steiner points in mind, the following lemma is worth noting:

Lemma 3.1. *There are arbitrarily large sets of points that, without the use of Steiner points, admit neither pseudo-even nor pseudo-odd triangulations.*

Proof. Let P be a set of points like the one shown to the left in Figure 3.1 where the size of the convex polygon Q shown in gray is $\equiv 1 \mod 3$. Let $v \in P$ be the only point not in Q. It is clear that in *any* triangulation of P, point v must be adjacent to *every* vertex of Q, that is, without a triangulation of Q, every vertex of Q has degree three.

Now, it is well known that *every* triangulation of a polygon has at least two "ears", *i.e.*, a triangle formed by three consecutive vertices of the polygon. This means that, regardless of what triangulation of Q we choose, there will always be a vertex of Q whose adjacencies are only its two neighbors in Q and v. Thus no pseudo-even triangulation of P exists. See to the right in Figure 3.1.

To show that P does not admit a pseudo-odd triangulation either it suffices to show that regardless of what triangulation of Q we choose, there will always be at least

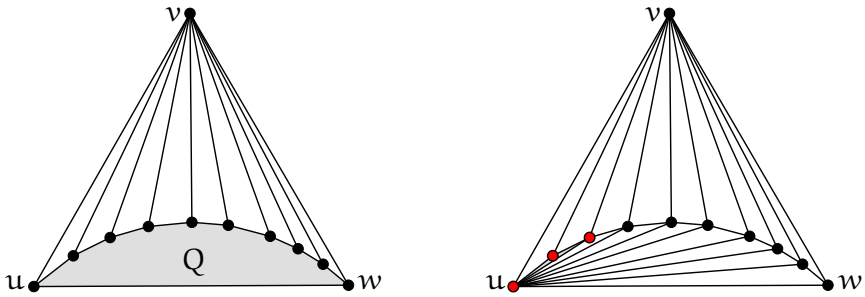

Figure 3.1 – To the left we have a configuration in which all shown adjacencies are forced, and it accepts neither pseudo-even nor pseudo-odd triangulations. To the right we show in red one of the ears of the shown triangulation of Q.

one interior point of P having even degree. It is not hard to verify that in an even triangulation, the size of the outer face must be $\equiv 0 \mod 3$, a proof can be found in [30]. Since $|Q| \equiv 1 \mod 3$, then Q does not admit an even triangulation, so in *every* triangulation of Q there will be at least two vertices of odd degree, there must be an even number of them. So assume that there is a triangulation of Q in which the only two vertices of odd degree are the two neighbors u, w of v in $\mathcal{CH}(P)$. Thus we could add a point p outside $\mathcal{CH}(P)$, below the edge uw, and add the adjacencies pu, pw. This implies that the set of points $Q' = Q \cup \{p\}$ has an even triangulation, where upw is an ear. But $|Q'| \equiv 2 \mod 3$, which clearly contradicts the necessary condition on the size of the outer face of an even triangulation. Therefore, in any triangulation of Q there must be at least one interior point $q \in P$ of odd degree. The force adjacency qv implies that q gets even degree in a triangulation of P, which is what we wanted to prove. ■

3.1 Our contribution

By Lemma 3.1 the use of Steiner points is sometimes *necessary* if we insists in constructing any of the four kinds of triangulations mentioned before (even, pseudo-even, odd, pseudo-odd). The relevant issue now is not whether we can construct the triangulations we are interested in, but rather with how many Steiner points can we achieve such constructions, the less, the better. The results we are going to show are the following:

Theorem 3.1. *Let $P \subset \mathbb{R}^2$ be a set of n points in general position where k of them are interior points. Then a set $S = S(P)$ of* interior *Steiner points of size at most $\lfloor \frac{k+2}{3} \rfloor + 2$ can always be constructed such that $P \cup S$ accepts a pseudo-even triangulation.*

Theorem 3.2. *Let P be as before. Then a set $S = S(P)$ of* interior *Steiner points of size at most $\lfloor \frac{n+1}{3} \rfloor + 1$ can always be constructed such that $P \cup S$ admits an even triangulation.*

Theorem 3.3. *Let* $P \subset \mathbb{R}^2$ *be a set of* n *points in general position where* k *of them are interior points. Then a set* $S = S(P)$ *of* interior *Steiner points of size at most* $\lfloor \frac{k}{3} \rfloor + c$*, with* c *a positive constant, can always be constructed such that* $P \cup S$ *accepts a pseudo-odd triangulation.*

Theorem 3.4. *Let* P *be as before. Then a set* $S = S(P)$ *of* interior *Steiner points of size at most* $\lfloor \frac{n-1}{3} \rfloor + c$*, with* c *a positive constant, can always be constructed such that* $P \cup S$ *admits an odd triangulation.*

The proofs of all theorems will be constructive. The rest of the chapter is divided as follows: In Section 3.2 we start with some preliminaries. In Section 3.3 and 3.4 we show algorithms that imply Theorems 3.1, 3.2, and Theorems 3.3, 3.4 respectively. We close the chapter in Section 3.5 we some observations and conclusions.

3.2 Pre-processing of P

Let us quickly recall that given a polygon $\mathcal{P}$, a vertex v of $\mathcal{P}$ is called *reflex* if the internal angle at v is larger than π. We will call it convex otherwise. Also, by a suitable rotation of P we can assume that the vertex v of $\mathcal{CH}(P)$ with the lowest y-coordinate is unique.

The following pre-processing of P will be done: Using v as a pivot we will sort each interior point of P by its angle with respect to v. Let $p_1, \ldots, p_k$, be a labeling, from left to right with respect to this angular order, of the interior points of P. Let p_0, p_{k+1} be the left and right neighbors of v on $\mathcal{CH}(P)$ respectively.

We construct a simple polygon $\mathcal{P}$ from $P \setminus \{v\}$ as follows: We add each edge $p_i p_{i+1}$, for $0 \leqslant i \leqslant k$. We call this the lower part of $\mathcal{P}$ and we will denote it by $L(\mathcal{P})$. Also, we consider the edges of $\mathcal{CH}(P) \setminus \{p_0 v, p_{k+1} v\}$ and we call this the upper part of $\mathcal{P}$, which will be denoted by $U(\mathcal{P})$, see Figure 3.2.

Next we will scan $L(\mathcal{P})$ from left to right and we will consider the largest polygonal chains that can be formed using consecutive convex vertices of $L(\mathcal{P})$. Note that for each such polygonal chain, the left and right vertices must be reflex vertices of $\mathcal{P}$, see Figure 3.3. Now, for each chain, we will make adjacent its two endvertices, thus forming convex polygons Q_j, with $j \geqslant 0$, from those convex polygonal chains. These convex polygons can be thought as big "ears" hanging from $L(\mathcal{P})$. We will triangulate the rest of $\mathcal{P}$ outside these Q_j's arbitrarily, see Figure 3.2. If there is no convex vertex of $\mathcal{P}$ in $L(\mathcal{P})$, then this arbitrary triangulation of $\mathcal{P}$ is a triangulation of all $\mathcal{P}$.

This is all the pre-processing that will be done. From here every algorithm will complete a triangulation of $\mathcal{P}$ in its own way.

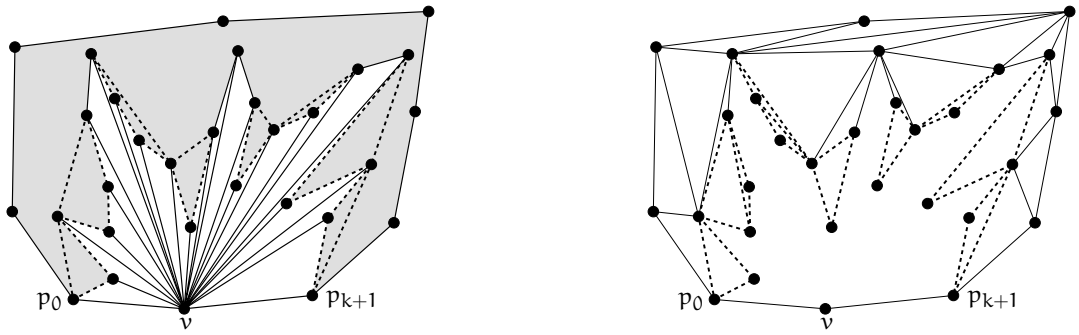

Figure 3.2 – To the left we have the polygon $\mathcal{P}$ on $n-1$ vertices in gray. The convex polygons formed by scanning $L(\mathcal{P})$ from left to right are shown dashed. Note that each pair of consecutive convex polygons shares at most one vertex. To the right we see a triangulation $T(\mathcal{P})$ of $\mathcal{P}$. The dashed edges are the only ones that are not arbitrary.

3.3 Even and pseudo-even triangulations

The following observation is very useful when working with 3-colorable triangulations:

Observation 3.1. Let T be a 3-colored triangulation with outer face C, not necessarily convex, and let u, v, w be three consecutive vertices of C. If the degree of v in T is even, then the color of u is different than the color of w. If the degree of v in T is odd, then both u, w have the same color.

We can now continue with the algorithm. Let us triangulate each Q_j, if any exists, as follows: Take its vertex with the lowest y-coordinate, breaking ties arbitrarily, and join it with straight-line segments to all other vertices of Q_j, in case that Q_j has more than three vertices. This is usually called a "fan" triangulation. These fan triangulations along the arbitrary triangulation outside the Q_j's complete a triangulation $T(\mathcal{P})$ of polygon $\mathcal{P}$.

It is well-known that triangulation $T(\mathcal{P})$ can be 3-colored, see [36], thus we will do it, and observe that the only colorless vertex is v, the one we used in the beginning to sort P angularly. Clearly, if a triangulation of P is 3-colorable, then it must be *at least* pseudo-even, thus our idea now is to complete a 3-colorable triangulation of P using $T(\mathcal{P})$ and its 3-coloring. So at all time we will use a 3-coloring as a measure of the correctness of our algorithm.

From this point on, our construction is done by case analysis. We will assume without loss of generality that the colors at our disposal are $\{0,1,2\}$. Note that as $T(\mathcal{P})$ is already 3-colored, if all the interior vertices of P are colored by only two colors, say $0, 1$, we could use color 2 for v without violating the 3-coloring of $T(\mathcal{P})$, and hence, using the

straight-line segments that connect v with each vertex of $L(\mathcal{P})$, we obtain a 3-colorable triangulation of P. Nevertheless, in general it is not going to happen that the interior vertices can be colored using only two colors, hence we need to do something else in such cases. We will proceed in a line-sweep fashion from p_0 to p_{k+1} with respect to the angular order around v.

Let us fix the color of v as the color of the smallest chromatic class in $L(\mathcal{P})$ using the 3-coloring of $T(\mathcal{P})$, and say that color is i[1] without loss of generality, $0 \leqslant i \leqslant 2$. Note that the points in $L(\mathcal{P})$ with color i are the ones causing trouble to complete the desired triangulation, hence we will handle those points depending on their kind in $\mathcal{P}$, namely if they are reflex or convex vertices of $\mathcal{P}$. We will keep the invariant that, by the time we are processing an interior point p_j, edge vp_j is present, and *all* interior points p_r, with $r < j$, have even degree already, except possibly for p_0. Also note that by this time, if the degree of p_j is odd it is because p_{j+1} has color i, due to Observation 3.1.

Let us start now with our case analysis, we will assume that we are currently processing the interior point p_j, $1 \leqslant j < k$, so again, we will assume that the edge vp_j is already present in the partial triangulation of P constructed so far. We have the following cases:

(1) Point p_{j+1} is of color i, just as v, and is a reflex vertex of $\mathcal{P}$. If points p_j and p_{j+2} are of different color then we can just add the edge p_jp_{j+2}, since p_{j+1} has already even degree in $T(\mathcal{P})$, see to the left in Figure 3.3. Thus we can add the edge vp_{j+2} and move to p_{j+2}. If on the other hand, p_j and p_{j+2} have the same color, we introduce one Steiner point s, below $L(\mathcal{P})$, of the third color different than the one of p_j and v, which will be adjacent to p_j, p_{j+1}, p_{j+2} and v, as shown in the middle in Figure 3.3. Thus we can again move to p_{j+2} and continue.

(2) Point p_{j+1} is of color i again but a convex vertex of $\mathcal{P}$. If p_j and p_{j+2} have the same color we proceed just as before, introducing one Steiner point s, below $L(\mathcal{P})$ and of the third available color, which will be again adjacent to p_j, p_{j+1}, p_{j+2} and v. We move to p_{j+2} and continue, see in the middle in Figure 3.3.

So let us assume that p_j and p_{j+2} have different colors, say w.l.o.g. $i+1$ and $i+2$ respectively. Note that, as p_{j+1} is a convex vertex of $\mathcal{P}$, it must be part of one of the convex polygons $Q_1, \ldots Q_r$, the big "ears", we constructed in the pre-processing phase. Let us denote this one convex polygon simply by Q, and its rightmost vertex by p_l, $l \geqslant j+2$, according to the angular order around v.

We have now the following sub-cases:

(2.1) Vertex p_{j+1} was used as a pivot in the fan triangulation of Q, see to the right in Figure 3.3. This in particular means that p_{j+1} is the only vertex of

[1] In the figures, unless otherwise stated, we will use colors $\{i, i+1, i+2\} = \{$black, red, blue$\}$, with arithmetic modulo 3.

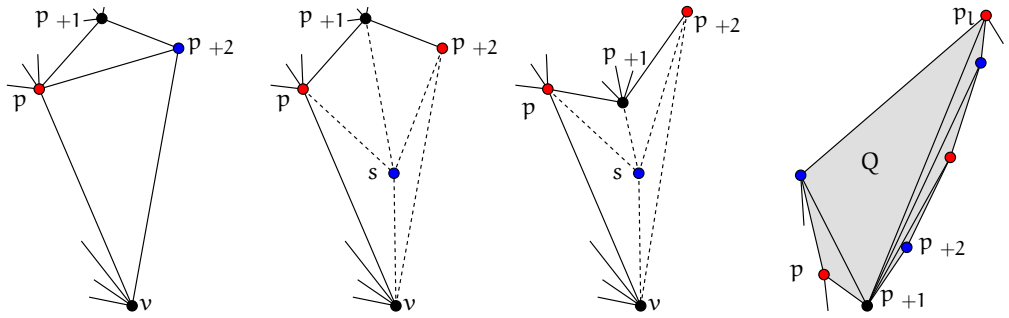

Figure 3.3 – The point p_j is currently being processed. Point p_{j+1} is of the same color i of v. If p_j and p_{j+2} have the same color, then one Steiner point suffices to be able to move to p_{j+2}. To the right the convex polygon Q is shown in gray. Point p_{j+1} is the pivot of the fan triangulation of Q.

color i in Q. If $l > j+2$ then we can re-triangulate Q by constructing the fan triangulation of Q with pivot at p_{l-1}, this changes the 3-coloring of Q only between p_{j+1} and p_{l-1}, the former receives color $i+2$ while the latter is the new unique vertex of Q of color i. Thus the only thing we did was to move the vertex of color i to the right, and hence we can join v to all vertices $p_{j+1}, \ldots, p_{l-2}$ using straight-line segments. If $l = j+2$ then Q is actually a triangle, and everything is still valid, v is connected to $p_{l-2} = p_j$, see to the left in Figure 3.4.

We now further distinguish between the following cases:

(2.1.1) Point p_l is of color $i+1$, p_{l+1} is of color i and p_{l+2} is of color $i+2$, see in the middle in Figure 3.4. We introduce two Steiner points s_1, s_2 of color $i+2, i+1$ respectively and we will make the following adjacencies: (1) s_1 gets adjacent to p_{l-1}, p_l, p_{l+1} and s_2, and (2) s_2 gets adjacent to $p_{l-2}, p_{l-1}, s_1, p_{l+1}, p_{l+2}, v$. Observe that these adjacencies can always be done without introducing any crossing. Moreover, note that with two Steiner points we complete the even degree of each point in the region $p_j, \ldots, p_{l+2}$ in which there were originally two points of color i. Thus we can move to p_{l+2} and continue.

(2.1.2) Point p_l and p_{l+1} as before, but p_{l+2} is of color $i+1$. We will proceed as before except that this time the adjacencies of s_1, s_2 are as follows: (1) s_1 gets adjacent to $s_2, p_{l-1}, p_l, p_{l+1}, p_{l+2}$ and v, and (2) s_2 gets adjacent to p_{l-2}, p_{l-1}, s_1 and v.

As before, we also introduce the adjacencies $p_{j+1}v, \ldots, p_{l-2}v$ and $p_{l+2}v$.

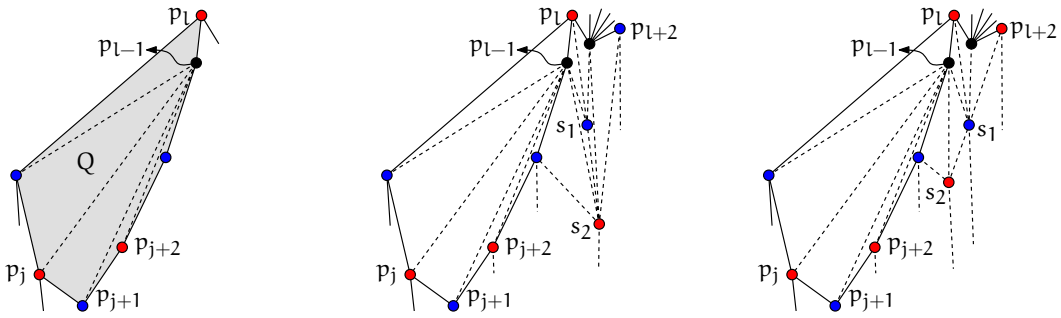

Figure 3.4 – If p_{j+1} was used as a pivot to triangulate a convex polygon that can be cut from $\mathcal{P}$, then we can use p_{l-1} as the new pivot without changing the color of p_j or anything to its left. Note that p_l must be necessarily a reflex vertex of $\mathcal{P}$. In the middle we see the final configuration in the case that p_{l+1} is of color i and p_{l+2} is of color $i+2$. To the right we see the final configuration when p_{l+1} is of color i and p_{l+2} is of color $i+1$.

We can again move to p_{l+2}. See to the right in Figure 3.4 for the final configuration.

(2.1.3) Point p_l as before but p_{l+1} is of color $i+2$ instead. Note that in this case, from p_j to p_{l+1} there is only one vertex of color i, namely p_{l-1}, thus we will introduce only one Steiner point s. Also observe that since p_l is a reflex vertex of $\mathcal{P}$ we can add the adjacency $p_{l-1}p_{l+1}$. Finally we make s adjacent to $p_{l-2}, p_{l-1}, p_{l+1}, v$, and we make, as before, v adjacent to $p_{j+1}, \ldots, p_{l-2}$ and p_{l+1}. See to the left in Figure 3.5 for the final configuration.

The following three cases are complementary:

(2.1.4) Point p_l is of color $i+2$, p_{l+1} is of color i and p_{l+2} is of color $i+1$.

(2.1.5) Point p_l and p_{l+1} as before and p_{l+2} is of color $i+2$.

(2.1.6) Point p_l as before and p_{l+1} is of color $i+1$.

However, these cases are the symmetric cases of (2.1.1), (2.1.2) and (2.1.3) respectively, where p_l gets the other possible color, and thus the constructions get shifted colors. The solution, as can easily be verified, is also of shifted colors. The two last figures of Figure 3.4 and the first one of Figure 3.5 can be used as reference if we shift their colors.

(2.2) In this case p_{j+1} of color i was not used as a pivot in the fan triangulation of Q. This means that p_j and p_{j+2} are adjacent, since they are part of Q. So

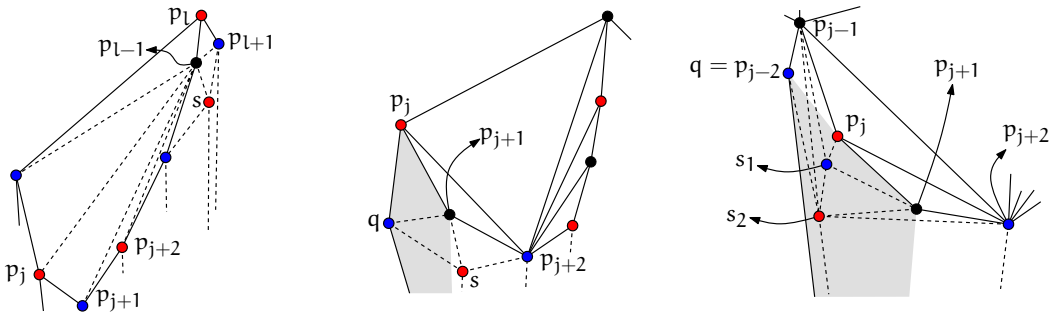

Figure 3.5 — To the left we see the final configuration in the case that p_{j+1} was a pivot of color i and $p_{\iota+1}$ is of color $i+2$. In the middle and to the right we have that, if p_{j+1} of color i was not a pivot and its neighbors have different color from each other, then one of them must necessarily be a pivot, p_{j+2} in this case. So we have to go back and remove some adjacencies that will allow us to introduce the Steiner points appropriately. Quadrilateral $\square$ is shown in gray.

we have two cases depending on whether p_j or p_{j+2} is the pivot in the fan triangulation of Q.

(2.2.1) Point p_j is the pivot in Q. Then consider the points two positions ahead, that is p_{j+3}, p_{j+4}. Let us assume w.l.o.g. that they exist, otherwise $p_{j+2} = p_{k+1}$, which is the right neighbor of v on $\mathcal{CH}(P)$, and p_j, p_{j+1}, p_{j+2} are the last three points that the algorithm will process. It is then easy to verify that two interior Steiner points suffice to finish.

Now, consider the pattern of colors of $p_j, \ldots, p_{j+4}$. The first three colors are fixed $i+1, i, i+2$. If we see $i, i+1$ next, then the pattern matches what we see in case (2.1.4)[II], which we know how to solve using two Steiner points, but we would also get rid of two points of color i. If we see next $i, i+2$, then we see the same as in case (2.1.5). If p_{j+3} is of a color different than i then it must necessarily be of color $i+1$, since p_{j+2}, of color $i+2$, and p_{j+3} are adjacent. But since p_j, of color $i+1$, is the pivot of Q, then p_{j+3} cannot be a vertex of Q, so p_{j+2} is actually a reflex vertex of $\mathcal{P}$, and we would find ourselves seeing what we see in case (2.1.6).

(2.2.2) Point p_{j+2} is the pivot in Q. This case is trickier since for the points $p_j, \ldots, p_{j+4}$ we could see the pattern of colors $i+1, i, i+2, i+1, i$, which we do not know how to solve locally using only two Steiner points. What we will do is to not look ahead but to see behind.

[II] Which in turn is the symmetric case of (2.1.1).

Since the edge $p_j v$ is currently in the triangulation being built, there is *exactly* one triangle using this edge. Let $q \notin \{p_j, v\}$ be the third vertex of such triangle. Note that q lies to the left of the edge $p_j v$ and hence it already has even degree, moreover, the color of q is $i+2$, since p_j, v have color $i+1, i$ respectively. Now we have the following two cases:

- Vertex q is a Steiner point, or the quadrilateral $\square = q, p_j, p_{j+1}, v$ is convex. Let us consider only the case that $\square$ is convex, if that is not the case then q is a Steiner point and it can be moved as pleased to make $\square$ convex without affecting anything.

  We will flip the edge $p_j v$ for the edge $q p_{j+1}$ and introduce one Steiner point s of color $i+1$ that will be adjacent to q, p_{j+1}, p_{j+2}, v, see in the middle of Figure 3.5.

- If q is not a Steiner point *and* $\square$ is non-convex, then it is not hard to see that the only possibility is $q = p_{j-2}$, and p_{j-1} must then be a reflex vertex of $\mathcal{P}$ of color i. Note then that the edge $e = p_{j-2}p_j$ must be present in the triangulation, by case (1), and thus p_{j-1} is adjacent to no Steiner point. Hence we will remove e and we will introduce one Steiner point s_1 of color $i+2$ that is adjacent to $p_{j-1}, p_j, p_{j+1}, s_2$, where s_2 is another Steiner point of color $i+1$ that is adjacent to $p_{j-2}, p_{j-1}, s_1, p_{j+1}, p_{j+2}, v$. We can now move to p_{j+2} and continue. See to the right in Figure 3.5.

Note that the color i of v was chosen as the color of the smallest chromatic class in $L(\mathcal{P})$, so its cardinality can be at most $\lfloor \frac{k+2}{3} \rfloor$. Now, it is important to observe that we assumed that the point p_j that we process is neither p_0 nor p_{k+1} of $\mathcal{CH}(P)$, since the algorithm started with $j \geqslant 1$. So it could happen that at least one of those extreme points is of the same color i of v, which would give a conflict in the 3-coloring of the triangulation we are constructing. Let us see how can we deal with this kind of situation. Assume without loss of generality that p_0 is of color i. Then, before start processing the first interior points p_1, introduce one Steiner point v' inside $\mathcal{CH}(P)$, so close to v that the angular order $p_1, \ldots, p_k$ w.r.t. v is also kept by v'. This Steiner point v' will replace v in the algorithm, so it will get color i as well. Now symbolically delete v and run the algorithm. When the algorithm ends we will still have the conflict of the monochromatic edge $p_0 v'$, we could simultaneously have the same conflict with edge $v' p_{k+1}$.

Put back v and remove the conflicting edges from the construction. We will proceed depending on what v sees, having the edges of the construction as obstacles. That is, if edge $p_0 v'$ is the only one with conflicts, then v sees the polygonal chain p_0, p_1, v', p_{k+1}, see to the left in Figure 3.6. If the edge $v' p_{k+1}$ also has conflicts then v sees the polygonal chain $p_0, p_1, v', p_k, p_{k+1}$, see to the right in Figure 3.6. The solution will

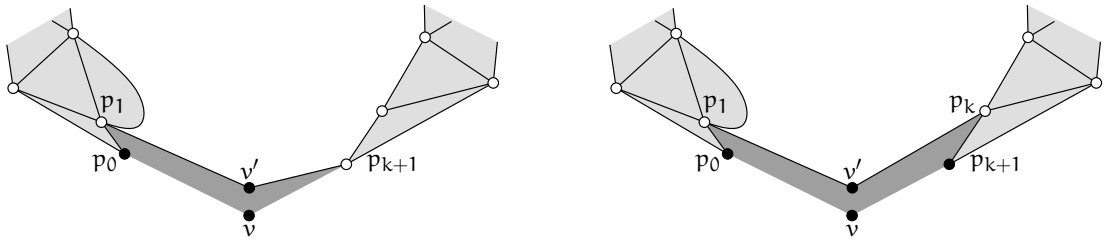

Figure 3.6 – Polygon $\mathcal{P}$ shown in light gray. In the figures color i = black, and the color white means that those points are somehow 3-colored without conflicting with the black points. The visibility region of v is shown in dark gray.

depend on whether p_0v' is the only conflict or not, and whether v' has even degree, in the triangulation constructed by the algorithm, or not. The four cases, shown in solid lines, and their solutions, shown with dashed lines, can be seen in Figure 3.7. Observe that at most one more Steiner point s is used for the solutions, and that v', s are both interior Steiner points. So s can be charged to p_0, which is one of the $\lfloor \frac{k+2}{3} \rfloor$ points of $L(\mathcal{P})$ of color i, and v' would be simply *one* interior Steiner point that cannot be charged to anything.

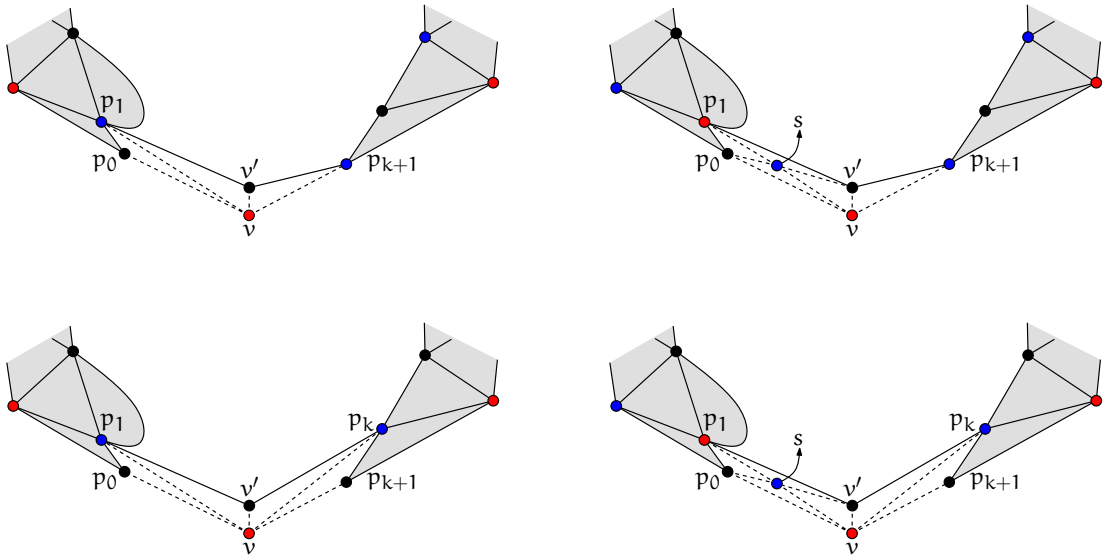

Figure 3.7 – Polygon $\mathcal{P}$ shown in gray. On the top part we have the solution for the cases where p_0v' is the only conflict and the degree of v' is odd, left, or even, right. Below we have the solutions for the case when both edges $p_0v', v'p_{k+1}$ are in conflict and the degree of v' is odd, left, or even, right. In the figures color i = black.

Finally, in the analysis of cases (2.1.1) to (2.1.6) we always assumed that point p_{l+1} always existed. This might not always be the case since we could have $p_l = p_{k+1}$, but in this case we could safely assume that $p_{l+1} = v$ or $p_{l+1} = v'$, depending on conflicts on $\mathcal{CH}(P)$, so the second Steiner point we introduce in those cases is also an interior Steiner point that cannot be charged to anything. Hence the construction used at most $\lfloor \frac{k+2}{3} \rfloor + 2$ interior Steiner points, and Theorem 3.1 follows.

3.3.1 Extension to even triangulations

Theorem 3.1 guarantees a 3-colorable triangulation, which will be at least pseudo-even, but it might not necessarily be completely even, and this is because when we choose an arbitrary triangulation for a part of $\mathcal{P}$, some vertices of $\mathcal{CH}(P)$ might get odd degree. Thus in order to construct an even triangulation we have to do some more work.

As we mentioned before, in an even triangulation the size of the outer face must be multiple of three. Thus the first thing we will do, if necessary, is to complete $\mathcal{CH}(P)$ using at most two Steiner points so that we fulfill $|\mathcal{CH}(P)| \equiv 0 \mod 3$.

Let v again the *unique* vertex of $\mathcal{CH}(P)$ with the smallest y-coordinate, and sort all $P \setminus \{v\}$ angularly, from left to right, around v. Let $p_1, \ldots, p_{n-1}$ be the points of $P \setminus \{v\}$ in this sorted order.

The main idea behind the construction is to enclose P in a bigger polygon $\mathcal{Q}$ so that all $p_0, \ldots, p_{n-1}$ are interior points, and then run the algorithm of Theorem 3.1, which will guarantee that all $p_0, \ldots, p_{n-1}$ will have even degree. The construction is done in such a way that $\mathcal{CH}(P)$ appears in the construction, at the end we can complete the degree of v to an even degree, if necessary, and such that by removing $\mathcal{Q}$ we do not destroy any parity, since $\mathcal{Q}$, upon deletion, will take with it an even number of adjacencies per "affected" point of $\mathcal{CH}(P)$, so the degree of those affected points will remain even at the end. So let us jump to the actual construction.

Enclose $\mathcal{CH}(P)$ in a bigger polygon $\mathcal{Q}$ with the following properties: (1) The size of $\mathcal{Q}$ is $|\mathcal{CH}(P)| + 1$, (2) The vertex of $\mathcal{Q}$ with smallest y-coordinate is also v and is unique, (3) The polygon $\mathcal{C}$ formed by $\mathcal{Q} \cup \mathcal{CH}(P)$ can be triangulated using a "zig-zag" starting at v. That is, if we denote the vertices of $\mathcal{CH}(P)$ in clockwise order starting with v by $q_1 = v, q_2, \ldots, q_k$, with $k \geqslant 3$ and $k \equiv 0 \mod 3$, and we denote the vertices of $\mathcal{Q}$ the same way by $q'_1 = v, q'_2, \ldots, q'_k, q'_{k+1}$, then the "zig-zag" is $q_1 = q'_1 = v, q'_2, q_2, q'_3, \ldots, q'_k, q_k, q'_{k+1}, v = q'_1 = q_1$, which along the edges of $\mathcal{C}$ complete a triangulation of $\mathcal{C}$ itself. See to the left in Figure 3.8.

This construction can always be achieved and has the following properties: (1) All points of $P \setminus \{v\}$ lie in the interior of $\mathcal{Q}$, (2) In the "zig-zag" triangulation of $\mathcal{C}$, every

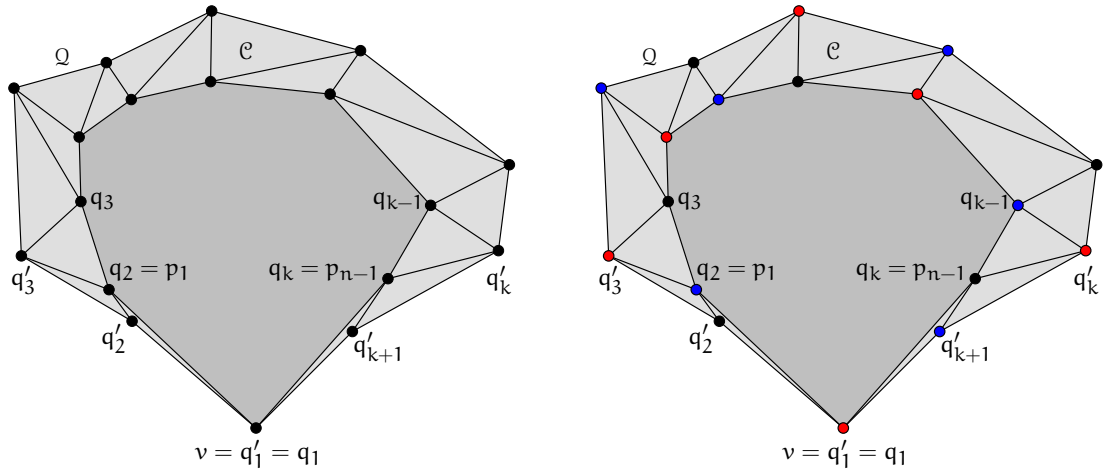

Figure 3.8 – To the left: The polygon $\mathcal{Q}$ is the outer face of the construction shown. Observe that it does not have to necessarily be convex. The convex hull of P, $\mathcal{CH}(P)$, is shown in dark gray, and $\mathcal{C}$ is shown in light gray, along with its zig-zag triangulation. To the right: The particular 3-coloring of the zig-zag triangulation of $\mathcal{C}$ using colors $\{0,1,2\} = \{\text{black}, \text{blue}, \text{red}\}$.

vertex of $\mathcal{CH}(P)$, and every vertex of $\mathcal{Q}$, except for q'_2, q'_{k+1}, has even degree, (3) In the 3-coloring of the triangulation of $\mathcal{C}$ that starts with color 0 at $q'_2 \in \mathcal{Q}$, and color 1 at $q_2 \in \mathcal{CH}(P)$, we have that $q_k \in \mathcal{CH}(P)$ receives color 0, vertex $q'_{k+1} \in \mathcal{Q}$ receives color 1, and v receives color 2. See to the right in Figure 3.8.

Now, pre-process $\mathcal{Q} \cup \mathcal{P}$ as explained in § 3.2. This pre-processing will construct in a first step the polygon $\mathcal{P}$ used in the algorithm of Theorem 3.1. The reader should see that this time $\mathcal{P}$ is the polygon whose upper part $U(\mathcal{P})$ is $\mathcal{Q} \setminus \{vq'_2, q'_{k+1}v\}$, and whose lower part $L(\mathcal{P})$ is formed by the adjacencies p_ip_{i+1}, for $1 \leqslant i \leqslant n-2$, along with the adjacencies q'_2q_2, $q'_{k+1}q_k$. In a second step, the "ears" formed by consecutive convex vertices of $L(\mathcal{P})$ will be computed. Recall that this "ears" are the convex polygons Q_j's in the pre-processing phase. Here, by suitably locating $q'_2, q'_{k+1} \in \mathcal{Q}$ we can assume that both $q_2, q_k \in \mathcal{CH}(P)$ are reflex vertices of $\mathcal{P}$, see Figure 3.9. Moreover, observe that *any* other vertex $q_i \in \mathcal{CH}(P)$, $2 < i < k$, must also be a reflex vertex of $\mathcal{P}$. This implies that the convex polygons Q_j's are contained in $\mathcal{CH}(P)$. This is important because in a third step of the pre-processing we compute a arbitrary triangulation of $\mathcal{P}$ minus those Q_j's. At this point we will choose the zig-zag triangulation of $\mathcal{C}$ as part of this "arbitrary" triangulation, adding other arbitrary edges inside $\mathcal{CH}(P)$ if the zig-zag triangulation of $\mathcal{C}$ does not complete a triangulation of $\mathcal{P}$ minus the Q_j's, the important thing here is that the zig-zag triangulation of $\mathcal{C}$, which contains $\mathcal{CH}(P)$, appears.

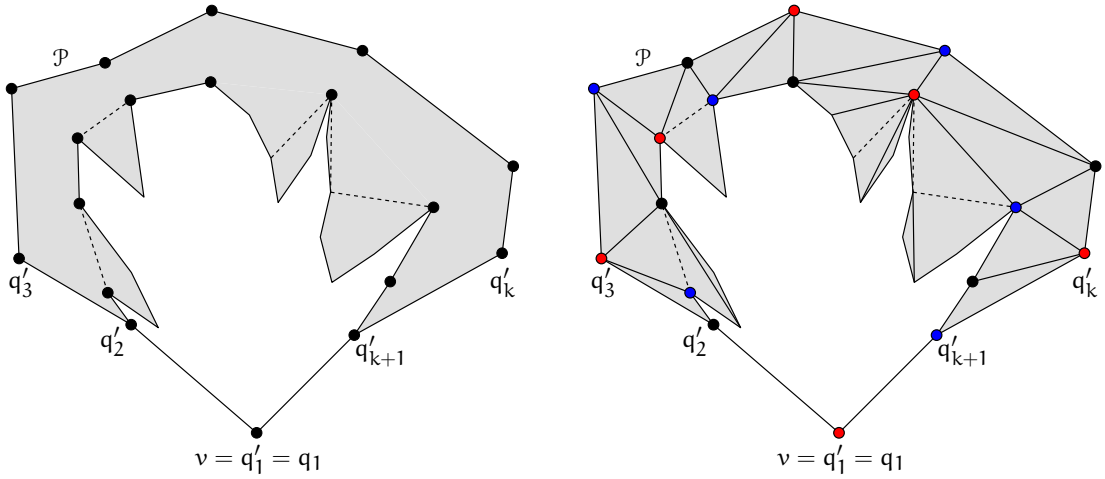

Figure 3.9 – Polygon $\mathcal{P}$ shown in gray. The dash lines delimit the "ears" that are constructed by the algorithm. They are contained in $\mathcal{CH}(P)$ since every vertex of $\mathcal{CH}(P)$ is a reflex vertex of $\mathcal{P}$. To the left we can see a whole 3-colored triangulation of $\mathcal{P}$, where the zig-zag triangulation of $\mathcal{C}$ appears. The 3-coloring is extended from the 3-coloring of $\mathcal{C}$.

If we now execute the algorithm of Theorem 3.1, the first thing it will do is to compute a fan triangulation of each Q_j, if any exists, see to the right in Figure 3.9. This will complete a triangulation $T(\mathcal{P})$ of polygon $\mathcal{P}$. The second thing it will do is to compute a 3-coloring of $T(\mathcal{P})$. Clearly we can extend the particular 3-coloring of the zig-zag triangulation of $\mathcal{C}$, explained before, to a 3-coloring of $T(\mathcal{P})$. Here again the important thing is that this particular 3-coloring of the triangulation of $\mathcal{C}$ appears. The third thing the algorithm will do is to re-color v with the color of the smallest chromatic class of the 3-coloring of $T(\mathcal{P})$. Here we have two cases:

- Vertex v stays of color 2. Observe that since $q'_2, q'_{k+1} \in \mathcal{Q}, q_2, q_k \in \mathcal{CH}(P)$, the neighbors of v in $\mathcal{Q}$ and $\mathcal{CH}(P)$ respectively, have colors 0 and 1, then v does not create any conflict with the 3-coloring of $T(\mathcal{P})$. Thus we will keep the adjacencies q'_2v, q_2v, vq_k and vq'_{k+1}, and we will execute the algorithm to the end. Since the colors of $q'_2, q'_{k+1}, q_2, q_k, v$ are never changed by the algorithm, we end up having a 3-colored triangulation, where v also has even degree, that uses at most $\lfloor \frac{n+1}{3} \rfloor + 1$[III] interior Steiner points. Note that these interior Steiner points are also interior w.r.t. $\mathcal{CH}(P)$, since the algorithm of Theorem 3.1 would introduce the Steiner points inside $\mathcal{Q}$, but strictly below the lower part $L(\mathcal{P})$ of $\mathcal{P}$. Since the adjacencies q'_2v, q_2v, vq_k and vq'_{k+1} are present in the output triangulation, the Steiner points fall strictly inside $\mathcal{CH}(P)$.

[III] It is not $\lfloor \frac{n+1}{3} \rfloor + 2$ this time since there is no conflict in $\mathcal{CH}(P)$.

The reader can verify that if we now remove $\mathcal{Q} \setminus \{v\}$, along with all the adjacencies that it takes with it, we are left with the desired even triangulation.

- Vertex v changes color. Then we will assume without loss of generality that v gets color 0. If v got color 1 we would have a symmetric conversation. Remember that by the particular 3-coloring of $\mathcal{C}$ we have that $q'_2, q'_{k+1} \in \mathcal{Q}$ have colors 0,1 respectively, and $q_2, q_k \in \mathcal{CH}(P)$ have color 1, 0 respectively, see Figure 3.8. Thus the algorithm will introduce the Steiner point v' of color 0, to take the place of v, and will symbolically delete v. This time we can charge v' to q'_2 which is one of the $\lfloor \frac{n+1}{3} \rfloor$ points of $L(\mathcal{P})$ of color 0. Now introduce the adjacency $v'p_{n-2}$ and execute the algorithm until it finishes. There, point p_{n-2} would be the last processed point. So we arrive at the configuration shown in the left upper corner of Figure 3.10.

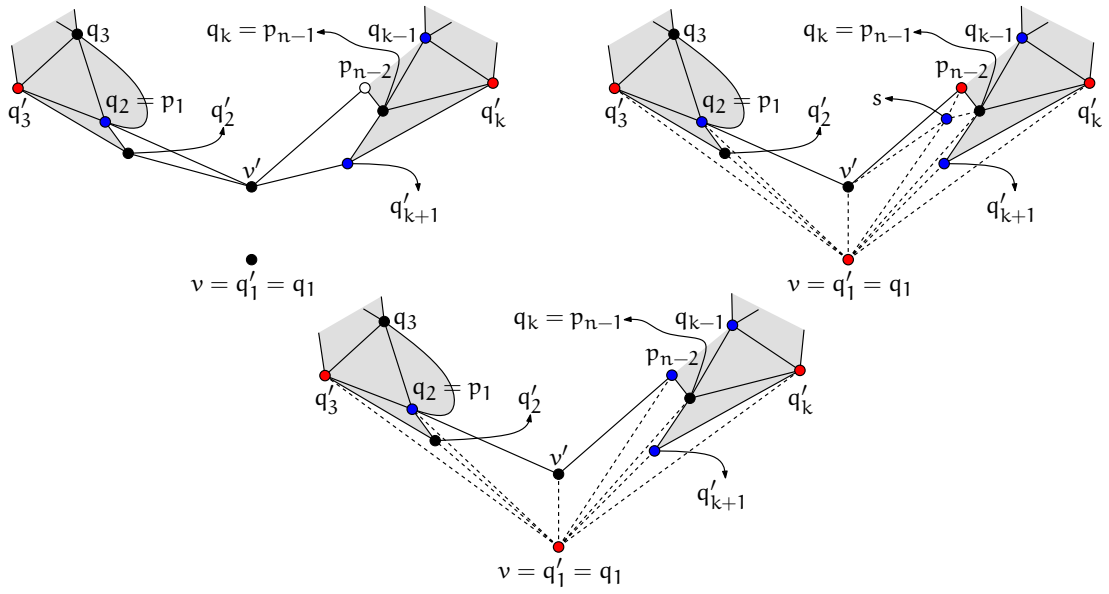

Figure 3.10 – Polygon $\mathcal{P}$ is shown in light gray. Using colors $\{0,1,2\} = \{\text{black}, \text{blue}, \text{red}\}$, if we made as if point p_{n-2} was the last point, we arrive at the configuration shown in the left upper corner. The white color of point p_{n-2} means that we do not care about its real color at this time. If we put v back, and we color it with 3, we can add the dashed adjacencies shown in the middle and the right upper corner, depending on the actual color of p_{n-2}. The color of v will conflict with the color of $q'_3, q'_k \in \mathcal{Q}$, but this is not a problem since in the end we will remove $\mathcal{Q} \setminus \{v\}$.

In this configuration we necessarily have that the degree of $q_2 = p_1 \in \mathcal{CH}(P)$ is odd, since it is adjacent to $q'_2, v' \in \mathcal{Q}$ which have both color 0. The degree of p_{n-2} is also odd since it is adjacent to $v', q_k = p_{n-1} \in \mathcal{CH}(P)$, and both are colored 0 as well. So the only options we have now are whether v' has even or odd degree

between q_2 and p_{n-2}. This is equivalent to consider whether p_{n-2} is of color 2 or 1 respectively. Those two configurations are shown in the right upper corner, and in the middle respectively, in Figure 3.10 in solid.

Now let us put v back and we will finish the construction as shown in Figure 3.10 with dashed lines. We might require the use of another Steiner point s, as shown to the right in the upper corner of the same figure, which can be charged to $q_k = p_{n-1}$. In the end v gets even degree as well. It is not hard to verify that the total number of interior Steiner points is again at most $\lfloor \frac{n+1}{3} \rfloor + 1$, that all Steiner points are interior to $\mathcal{CH}(P)$, and that the removal of $Q \setminus \{v\}$ does not destroy any parity. Hence Theorem 3.2 follows.

3.4 Pseudo-odd and odd triangulations

Working locally with (pseudo-)odd triangulations is slightly easier. The following observation was already pointed out in [65]:

Observation 3.2. Let $\triangle$ be a triangle in a triangulation. Then at most seven interior Steiner points suffice to obtain an odd-triangulation of $\triangle$.

The way this odd triangulation of $\triangle$ is obtained is shown in the following figure:

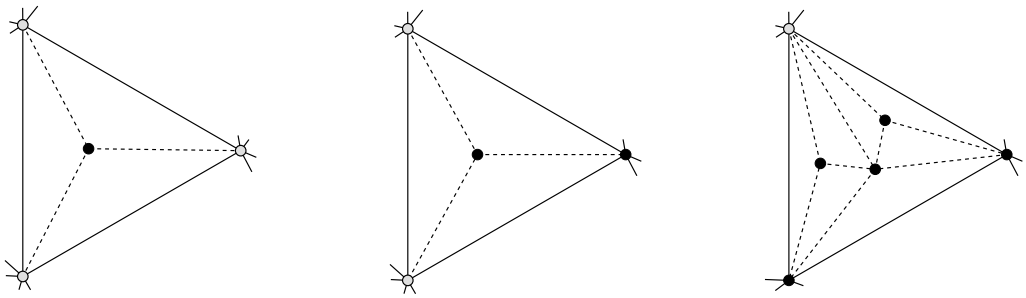

Figure 3.11 – All interior points are Steiner points. Gray vertices are of even parity before the introduction of Steiner points, and all black vertices on the boundary of $\triangle$ are of odd parity. The middle case spawns two sub-problems, of the kind shown to the right, by introducing one Steiner point. This gives in total $1 + 3 + 3 = 7$ Steiner points for an odd-triangulation of $\triangle$.

Observe that, in general, no similar statement can be done for (pseudo-)even triangulations.

For simplicity we will right now focus on pseudo-odd triangulations only. The algorithm to construct them is essentially the same as for pseudo-even triangulations, however, this time we will not be able to use a 3-coloring as guide. That 3-coloring played an important role in the upper bound on the number of interior Steiner points that we used: We did not have to explicitly track whether we introduced at most one Steiner point every three interior points, but we just had to take care of solving conflicts with the coloring as they appeared. The 3-coloring then ensured such conflicts to appear at most one-third of the time. So for (pseudo-)odd triangulations we will have to explicitly take care of introducing at most one Steiner point per every three interior points.

The pre-processing of P will be again as explained in § 3.2. Let $v \in P$ be again the unique vertex with smallest y-coordinate, the one we use to sort the interior points angularly around. The pre-processing phase ends with the construction of the convex polygons Q_j's, the big ears hanging from the lower part $L(\mathcal{P})$ of the therein created polygon $\mathcal{P}$. For pseudo-even triangulations we would fan-triangulate those Q_j's, if any. For pseudo-odd we will choose different triangulations, the main idea is the following: If *every* vertex of a convex polygon Q_j is of even degree in a triangulation $T(\mathcal{P})$ of $\mathcal{P}$, then we could just directly make them adjacent to v and leave them all odd. Unfortunately this might not always be possible. We will try nonetheless to achieve something similar, and the following result shown in [4] will be very useful:

Lemma 3.2 (O. Aichholzer *et al.*). *Let Q be a convex polygon where each of its vertices has a parity assigned. Let p, q, r be any three consecutive vertices of Q. Then there exists a triangulation of Q that makes all vertices of Q happy with the possible exception of p, q, r.*

After the pre-processing there is an arbitrary triangulation of $\mathcal{P}$ minus the Q_j's. We will complete a triangulation $T(\mathcal{P})$ of $\mathcal{P}$ as follows: for each one of the Q_j's set the parity of its vertices appropriately so that we can apply Lemma 3.2 and make them all even in $T(\mathcal{P})$, with the possible exception of its last three vertices, w.r.t. the angular order of the interior points around v. The triangulations of those Q_j's, however, might change during the algorithm depending on what conflicts we encounter.

Having this particular triangulation $T(\mathcal{P})$ of $\mathcal{P}$ we will again scan $L(\mathcal{P})$ from the left to right, following the angular order $p_1, \ldots, p_k$ of the k interior points of P around v, so again p_0 and p_{k+1} are the neighbors of v in $\mathcal{CH}(P)$. We will again assume that by the time we are processing point p_j, the adjacency $p_j v$ is already present and every point p_r, $r < j$, is of odd degree in the current construction. Nevetheless, observe that since we want to add roughly $\frac{k}{3}$ interior Steiner points, processing p_j actually means processing p_j, p_{j+1}, p_{j+2}, so we have different cases depending on the local situation. That is, if we are currently stuck at p_j it is because its current degree is even, otherwise we could just continue. So the cases we have to study are the triples of parities:

$(e,e,e), (e,e,o), (e,o,e)$ and (e,o,o), where they correspond entry-wise to the current parities of (p_j, p_{j+1}, p_{j+2}), and e, o stand for even and odd respectively. It is very important to keep in mind that in the triple (p_j, p_{j+1}, p_{j+2}), the only point adjacent to v is p_j. Also, in *all* cases we will assume that p_{j+3} exists. If that is not the case then one of p_j, p_{j+1}, p_{j+2} is p_{k+1}, and we would find ourselves with a problem of constant size that can be solved using a constant number of interior Steiner points, due to Observation 3.2. We will now jump to the case distinction.

(1) (e,e,e). Regardless of whether p_j, p_{j+1}, p_{j+2} are reflex or convex vertices of $\mathcal{P}$, the configuration is shown in solid to the left in Figure 3.12 and its solution is shown dashed.

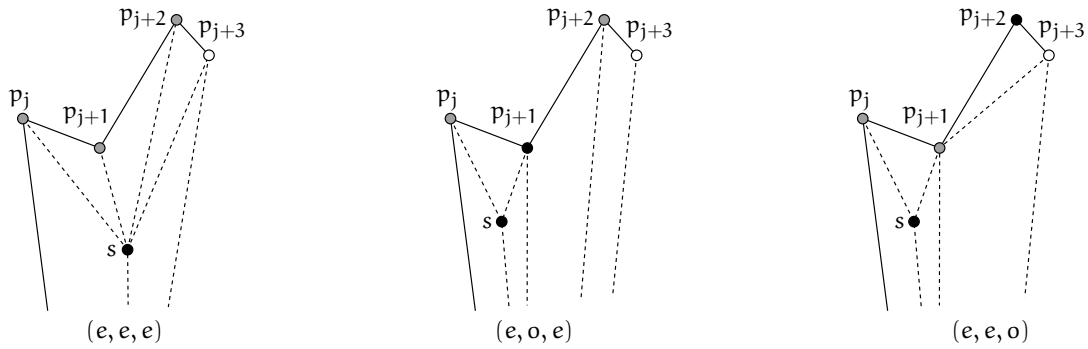

Figure 3.12 – In the figures, the colors represent the parity of the vertices *before* the adjacencies of the solution are added. Gray color means even degree, black color means odd degree, and white means that we do not necessarily take care of that point at this step. The original configurations are shown in solid black while their solutions are shown dashed. Point s is a Steiner point.

(2) (e,o,e). Regardless of whether p_j, p_{j+1}, p_{j+2} are reflex or convex vertices of $\mathcal{P}$, look the middle configuration of Figure 3.12.

(3) (e,e,o). If p_{j+2} is a reflex vertex of $\mathcal{P}$, then the situation is shown to the right in Figure 3.12.

If p_{j+2} is a convex vertex of $\mathcal{P}$, then p_{j+2} is part of some convex polygon Q we applied Lemma 3.2 on. Observe that Q cannot be a triangle, otherwise p_{j+2} would be the middle vertex and therefore p_{j+2} would have even degree, since its two neighbors in Q would be adjacent. Thus Q must have size at least four. This implies that p_j cannot be part of Q either, because otherwise we could safely assume that p_j, as any other vertex of Q to the left of p_j, is happy due to Lemma 3.2. This means that p_{j+1} is necessarily a reflex vertex of $\mathcal{P}$, and thus the leftmost vertex of Q. Finally, this all means that Q cannot actually have size larger than four, *i.e.*,

Q must necessarily be a convex quadrilateral. If Q had size larger than four we could safely assume that the degree of p_{j+2} in the triangulation of Q is even due to Lemma 3.2 and the fact that neither p_{j+1} nor p_{j+2} have been modified by the algorithm before. Therefore the situation is as pictured in the upper left corner of Figure 3.13 and its solution is shown right below.

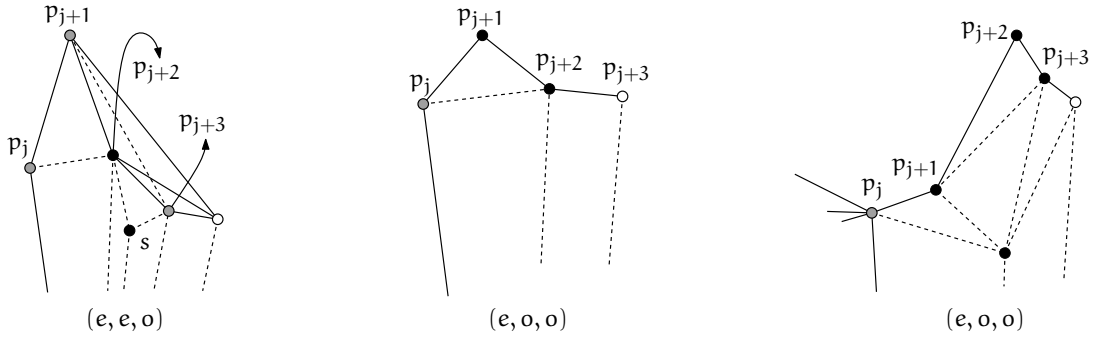

Figure 3.13 — The colors represent the parity of the vertices *before* the adjacencies of the solution are added. Gray color means even degree, black color means odd degree, and white means that we do not necessarily take care of that point at this step. The original configurations are shown in solid black while their solutions are shown dashed. Point s is a Steiner point.

(4) (e, o, o). This case is the hardest of all. If p_{j+1} is a reflex vertex of $\mathcal{P}$, then the situation is shown in the second figure in the upper part of Figure 3.13, and whose solution is shown right there dashed. If p_{j+1} is a convex vertex of $\mathcal{P}$, then observe that p_{j+2} cannot be a convex vertex as well, since otherwise $p_j, \ldots, p_{j+3}$ are part of the same convex polygon, and p_j would be happy due to Lemma 3.2. So p_{j+2} is a reflex vertex of $\mathcal{P}$. More, if the degree of p_{j+3} in $T(\mathcal{P})$ is odd then the solution is like shown to the right in Figure 3.13.

Thus we enter a case where p_{j+2} is a reflex vertex of $\mathcal{P}$ and p_{j+3} is of even degree in $T(\mathcal{P})$. This case cannot be solved locally, considering only $p_j, \ldots, p_{j+3}$, and there is more than one way we can deal with it. The way we will do it here is the following: We will go ahead until p_{j+6} and we will solve $p_j, \ldots, p_6$ with at most *two* Steiner points, one of them will be charged to the triple p_j, p_{j+1}, p_{j+2}, and the other to the triple $p_{j+3}, p_{j+4}, p_{j+5}$. At the end of the construction we will be left with the adjacency $p_{j+6}v$, which is where the algorithm will continue. So for all what follows we will assume that $p_{j+4}, \ldots, p_{j+6}$ exist, otherwise $p_{j+3} = p_{k+1}$, and we find ourselves with the last four points the algorithm would process. This can be solved with a constant number, larger than one, of Steiner points, which turns into a constant overhead overall.

Since we are assuming that p_{j+3} is currently even, we have four main sub-cases that depend on the pair of parities $(\cdot,\cdot)$ of (p_{j+4}, p_{j+5}) in $T(\mathcal{P})$. The configurations and solutions are pictured in Figure 3.14.

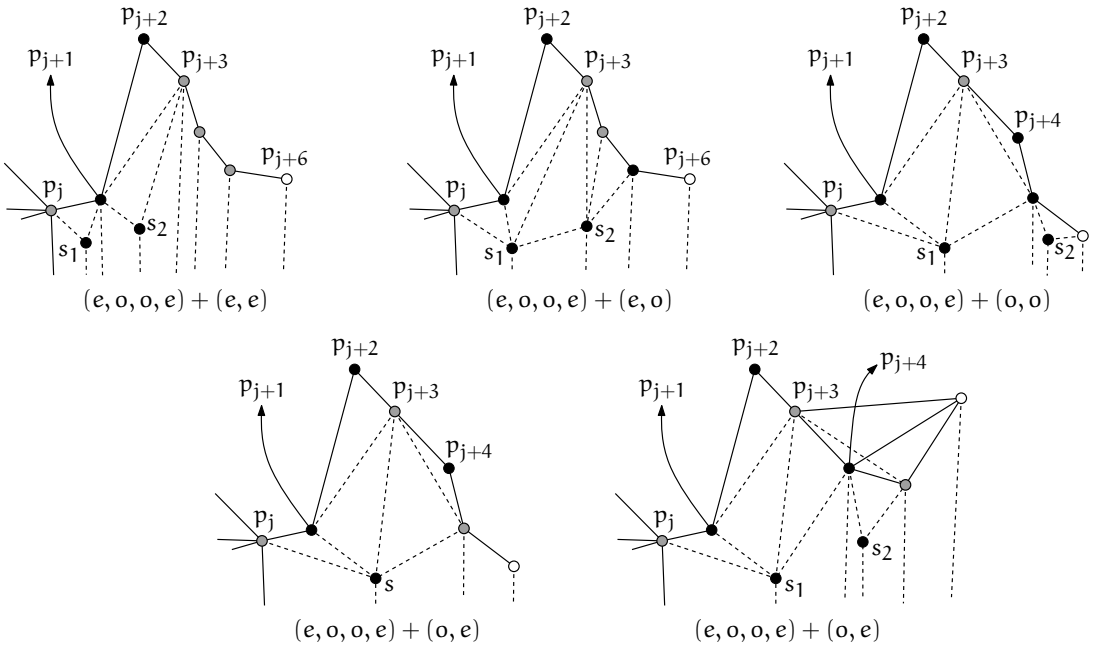

Figure 3.14 – The colors represent the parity of the vertices *before* the adjacencies of the solution are added. Gray color means even degree, black color means odd degree, and white means that we do not necessarily take care of that point at this step. The original configurations are shown in solid black while their solutions are shown dashed. Points s_1, s_2 are Steiner points.

(4.1) (e,e). The configuration is shown in the left upper corner of Figure 3.14 in solid and its solution is shown dashed. Observe that for the solution it does not play a role whether $p_{j+3}, p_{j+4}, p_{j+5}$ are reflex or convex vertices of $\mathcal{P}$.

(4.2) (e,o). The configuration is shown in the second figure on the upper part of Figure 3.14 in solid and its solution is shown dashed. Again, it does not matter whether $p_{j+3}, p_{j+4}, p_{j+5}$ are reflex or convex vertices of $\mathcal{P}$.

(4.3) (o,o). If p_{j+4} is a reflex vertex of $\mathcal{P}$, the configuration is shown in solid to the right on the upper part of Figure 3.14, and its solution is shown dashed.

Now let us argue that p_{j+4} cannot be a convex vertex. If p_{j+4} were a convex vertex it would be part of a convex polygon Q we applied Lemma 3.2 on in the beginning. This polygon Q cannot be a triangle, because then p_{j+4}

would be the middle point and it would then have even degree in $T(\mathcal{P})$. Also, Q cannot have at least five sides because by Lemma 3.2, again, p_{j+4} could be made of even degree in a triangulation of Q, without affecting the parity of p_{j+3} in the same triangulation. Thus the only case that kind of makes sense is that Q is a convex quadrilateral. So, there are two possible options for Q. Either $Q = p_{j+2}, p_{j+3}, p_{j+4}, p_{j+5}$ or $Q = p_{j+3}, p_{j+4}, p_{j+5}, p_{j+6}$. The former is not possible because p_{j+2} would be part of Q and the algorithm would have chosen in the beginning to leave p_{j+2} of even degree, instead of odd, using a triangulation of Q. This is achievable using Lemma 3.2. The latter is also not possible because although p_{j+3} is currently of even degree, p_{j+4}, p_{j+5} are of odd degree in the current triangulation of Q. Since p_{j+4}, p_{j+5} are the two middle vertices of Q, and Q has only two triangulations, then at least one of them would get even degree, and again, we would be discussing a different case. Therefore p_{j+4} must necessarily be a reflex vertex of $\mathcal{P}$.

(4.4) (o, e). If p_{j+4} happens to be a reflex vertex of $\mathcal{P}$, then the configuration is shown to the left on the lower part of Figure 3.14, and its solution is shown dashed.

If p_{j+4} is a convex vertex of $\mathcal{P}$, then its part of a convex polygon Q we applied Lemma 3.2 on in the beginning. By the same arguments as in case (o, o) we know that Q can be neither a triangle nor a convex polygon larger than four. Nevertheless, this time Q can indeed be the quadrilateral $p_{j+3}, p_{j+4}, p_{j+5}, p_{j+6}$. The other possibility for Q is in this case not possible either due to the same argument. Since Q is a convex quadrilateral and p_{j+4} is of odd degree in the current triangulation of Q, then the only possibility is that the diagonal $p_{j+4}p_{j+6}$ is present. The configuration is shown to the right on the lower part of Figure 3.14 in solid and its solution is shown dashed.

This concludes the case analysis.

From all cases discussed it is clear that the algorithm introduces at most one interior Steiner point per triple of interior points of P, and at the end it might require to brute-force a configuration of constant size, that due to Observation 3.2 can be solve using a constant number c of interior Steiner points as well. Hence overall the algorithm makes use of at most $\lfloor \frac{k}{3} \rfloor + c$ interior Steiner points and Theorem 3.3 follows.

3.4.1 Extension to odd triangulations

The extension of the algorithm of Theorem 3.3 to odd triangulations is now very easy. We again enclose the set of points P in a bigger polygon, just as we did for even triangulations. What the configuration looks like can be seen to the left in Figure 3.8 on page 41.

We then run the algorithm for pseudo-odd triangulations, and at the end the only thing that can happen is that the pivot $v \in P$ gets even degree. However, by Observation 3.2, pivot v can be made odd by locally adding a constant number of Steiner points in one of its adjacent triangles. Thus, in total, the number of used interior Steiner points is at most $\lfloor \frac{n-1}{3} \rfloor + c$, for some constant c. The odd triangulation is finally obtained by removing the bigger polygon enclosing P. Therefore Theorem 3.4 follows.

3.5 Conclusions

In this chapter we have presented algorithms that construct, with help of Steiner points, (pseudo-)even and (pseudo-)odd triangulations of a given set of points P. The number of Steiner points that the algorithms use is roughly $\frac{k}{3}$ for the pseudo variants, where k is the number of interior points of P, and roughly $\frac{n}{3}$ for the other cases. It is important to observe that our algorithms *do not* modify the convex hull of P, and therefore they preserve extent measures of P, such as diameter, width, among others. If we do not care about modifying $\mathcal{CH}(P)$, or about the position of the Steiner points, then the task is in general significantly easier. For example, *only two* Steiner points far away from $\mathcal{CH}(P)$ would suffice to construct a pseudo-even triangulation, say one at ∞ and the other at $-\infty$. Albeit being this construction possible, we do not know why it would be interesting to use it, since the output set of points does not look anything like the one that was given as the input.

We could believe that the techniques shown here could be pushed further to improve the overall number of Steiner points, say to go from one-third to one-sixth, but this will imply a significantly larger number of cases to analyze, we see this really as a secondary interesting improvement. What we really see as the primary open problems are the following:

- Is it possible to always construct (pseudo-)even or (pseudo-)odd triangulations of a given set of points P using only a *constant* number of (interior) Steiner points? In other words, how big is the lower bound on the number of (interior) Steiner points that are required to *always* construct such triangulations? We have failed so far trying to prove a (sub-)linear lower bound, which is the natural guess when working on these kind of problems.

- Moving away from using Steiner points, would it be possible to construct even or odd triangulations of a given set of points P where at most a constant number of points remain unhappy? This question was posed by Aichholzer *et al.* in [4]. In that same paper they proved a lower bound of roughly $\frac{n}{108}$ unhappy vertices when the assignment of parities is not uniform. Thus, as they pointed out, the interesting cases are all even and all odd.

Note that although these two question look similar, they might not be equivalent. If we are interested in an even triangulation of P and we construct one where a constant number of vertices remain unhappy, those unhappy vertices might be far from each other. This means that we might have to bring them together, at least in pairs, using Steiner points. We have to get rid of them of them in pairs, at least, since the number of odd vertices is always even. But to get them close to each other we might require more than a constant number of Steiner points. Thus the techniques to solve those two open problems might be rather different. We see a real challenge there.

CHAPTER 4

A SWEEP LINE ALGORITHM FOR COUNTING TRIANGULATIONS AND PSEUDO-TRIANGULATIONS

While triangulations require essentially no introduction, due to their many applications, pseudo-triangulations are way less known. Pseudo-triangulations were originally used in [66] for sweeping complexes, and in [23, 42] for ray-shooting. However, it was until a paper of Ileana Streinu appeared, see [81], that pseudo-triangulations really took off as a main research topic, due to their structural richness. In the same paper, [81], a particular kind of pseudo-triangulations was introduced, the so-called *pointed* pseudo-triangulations. In a pointed pseudo-triangulation *every* vertex is incident to an angle larger than π, and its characterization is very rich. The following is just a subset of equivalences found in [81]:

Theorem 4.1 (I. Streinu). *Let* G *be a straight-edge plane graph on a set of points* P*. The following properties are equivalent:*

- G *is a pointed pseudo-triangulation.*
- G *is a pseudo-triangulation having the* minimum *number of edges, and thus also the* minimum *number of pseudo-triangles.*
- *The set of edges of* G *forms a* maximal*, by inclusion, planar and pointed set of edges,* i.e.*, a set of edges whose union is crossing-free, and in which every vertex is incident with an angle larger than* π*.*

Pointed pseudo-triangulations have found interesting applications in robot arm motion planning, see [81], and have been the subject of extensive research, see the survey on pseudo-triangulations in [38], which is an excellent reference for most known results to date on pseudo-triangulations.

In this work we will be concerned only with pointed pseudo-triangulations, so we will drop the "pointed" part and we will only call them pseudo-triangulations. So, unless otherwise stated, our pseudo-triangulations are *always* pointed. No confusion shall arise.

Knowing what triangulations and pseudo-triangulations are, we can talk about the classes $\mathcal{F}_T(P)$ and $\mathcal{F}_{PT}(P)$ of *all* triangulations and *all* pseudo-triangulations of a given set of n points P respectively, and ask about their sizes, how large are they? We can actually think about two flavors of this question: (1) What is the *largest* or *smallest* they can get over all sets $P \subset \mathbb{R}^2$ of n points? or (2) Given P, what is the *exact* size of a desired class?

The first question mentioned above requires usually heavy mathematical machinery since the number of *combinatorially different* configurations of n points is too large to be explored by computer, see [40]. Thus, the first question is of rather theoretical flavor and it has actually spawned a large amount of research over almost 30 years, which started with the seminal work of Ajtai, Chvátal, Newborn and Szemerédi, where they showed that the number of *all* crossing-free structures on any set of n points on the plane can be at most 10^{13n}, see [7]. This bound implies that the size of *each* class of crossing-free structures on P can be upper-bounded by c^n, with $c \in \mathbb{R}$ depending on the particular class. Since then research has focused on fine-tuning c. For example, in the case of triangulations, the most popular in recent years, it is currently known that $2.4 \leqslant c \leqslant 30$, see [75] for the upper bound and [77] for the lower bound. Thus *every* set P of n points on the plane fulfills $|\mathcal{F}_T(P)| = \Omega(2.4^n)$ and $|\mathcal{F}_T(P)| = O(30^n)$. For the class of pseudo-triangulations not much is known. For example, it is known that c attains its minimum value for sets of points in convex position, *i.e.*, $c \geqslant 4$, see [3]. It is also known that $|\mathcal{F}_{PT}(P)| \leqslant 3^i|\mathcal{F}_T(P)|$, where i is the number of interior points of P, see [69].

As for the second question mentioned before, we always assume that we are given a set P of n points on the plane and we are interested in computing the exact values of $|\mathcal{F}_T(P)|, |\mathcal{F}_{PT}(P)|$, for example, the set of 32 red points presented in Figure 4.1, representing the State Capitals of Mexico, spans *exactly* 6 887 011 250 368 237 767 $\approx 3.8787^{32}$ triangulations.

The second question is thus of empirical flavor, and therefore algorithmic, since no closed-form formula is known, in general, for $|\mathcal{F}_T(P)|, |\mathcal{F}_{PT}(P)|$. It is then important to come up with methods (algorithms) that can compute their sizes efficiently. A first approach would be to produce *all* elements of the desired class, using methods for enumeration, see for example [14, 16, 15, 49], and then simply count the number of

Figure 4.1 – A set of 32 points representing the State Capitals of Mexico.

elements. This has the obvious disadvantage that the total time spent will be, at best, linear in the number of elements counted, which, by the first part, is always exponential in the size of the input. Thus, the crucial question is whether $|\mathcal{F}_T(P)|, |\mathcal{F}_{PT}(P)|$ can be computed faster, say, for starters, in time *sub-linear* in the number of elements counted. Currently this is only known for the super class of *all* crossing-free structures on the given set P of n points, see [71]. For the particular classes $\mathcal{F}_T(P), \mathcal{F}_{PT}(P)$ there are algorithms that seem to count faster than enumeration, see [2, 70, 6], but no theoretical runtime guarantees are known.

This and the next two chapters are fully devoted to the second question, namely, the algorithmic version of the problem of counting triangulations.

4.1 Our contribution

In this chapter we focus on counting the elements of $\mathcal{F}_T(P)$ and $\mathcal{F}_{PT}(P)$. We will only be concerned about algorithms with provable running times.

4.1.1 The result on counting triangulations

In order to state our results we will require some definitions, which for clarity we state first:

Definition 4.1 (Separating line). Let P be a non-empty set of points on the plane, and let l be a straight line such that $l \cap P = \emptyset$ but $l \cap \mathcal{CH}(P) \neq \emptyset$, then l will be called a *separating line* w.r.t. P.

Definition 4.2 (T-path). Given a non-empty set of points P on the plane, a triangulation T of P, and a separating line l w.r.t. P, a T-path of T w.r.t. l, denoted by $p(l, T)$, is defined as follows: (1) $p(l, T)$ is a chain of edges of T where every edge of $p(l, T)$ intersects l. (2) Starting and ending edges of $p(l, T)$ are two edges of $\mathcal{CH}(P)$ intersected by l. (3) The area bounded by two consecutive edges of $p(l, T)$ and l must be empty of points of P. See to the left in Figure 4.2 for an example of a T-path $p(l, T)$.

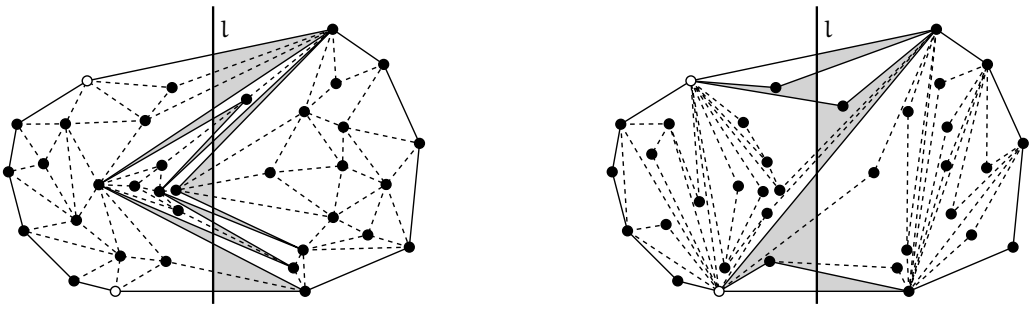

Figure 4.2 – To the left a T-path $p(l, T)$, shown in solid lines, of a triangulation T with vertex set P. To the right a PT-path $pt(l, S)$, shown also in solid lines, of a pseudo-triangulation S with vertex set P. The gray areas are the areas bounded by two consecutive edges of the paths and line l, which are empty of points of P.

T-paths were originally introduced by Oswin Aichholzer in 1999 in [2]. What makes them relevant is the following theorem, also presented in [2]:

Theorem 4.2 (O. Aichholzer). *Let P be a set of points and l a separating line w.r.t. P. Then the following holds: (1) For every triangulation T of P there always exists a T-path $p(l, T)$. (2) $p(l, T)$ is unique for T. (3) If T and T' are two triangulations of P, then $p(l, T)$ and $p(l, T')$ are either equal, or properly intersect each other,* i.e., *there are intersection points lying in the strict interior of their edges.*

Moreover, in the same paper, Aichholzer designed an algorithm to compute $|\mathcal{F}_T(P)|$ based on T-paths and the divide-and-conquer paradigm. His algorithm experimentally exhibited a running time sub-linear in the number of triangulations counted, that is, that algorithm was apparently faster than enumeration. A formal proof of this fact is, however, hard to obtain since it is not clear how to show that a single T-path appears in many triangulations, even on average. Nonetheless, the running time of Aichholzer's algorithm can be bounded by the number of sub-problems that it generates. Since the

algorithm is based on the divide-and-conquer paradigm, we can describe its running-time recurrence by $R(n) = 2t(n) \cdot R(n/2)$, where $t(i)$ denotes the number of T-paths encountered by the algorithm when i points are considered. If we can show that $t(i) \leqslant a^i$, for some positive constant a, we have that $R(n) \leqslant 2 \cdot a^n \cdot R(n/2)$, which gets solved to $O\left(a^{2n}\right)$. It is important to note here that $t = t(n)$ can become exponentially large, for example, Aichholzer showed that the convex polygon on n vertices has roughly $O\left(2^n\right)$ T-paths, and in [32] a configuration is shown that has $\Omega\left(2^{2n-\Theta(\log(n))}\right)$ T-paths, which is essentially 4^n, so the quadratic term in the running time of Aichholzer's algorithm becomes really expensive. The first contribution of ours that will be shown is the following theorem:

Theorem 4.3 (V. Alvarez, K. Bringmann, S. Ray). *Let P be a given set of n points on the plane. Then the exact value of $|\mathcal{F}_T(P)|$ can be computed in $O\left(n^3 \cdot t\right)$ time, and $O(t)$ space, where t is the largest number of T-paths the algorithm encounters when run on P. Moreover $t = O(9^n)$.*

Thus the running time of our algorithm for computing $|\mathcal{F}_T(P)|$, based on T-paths, can really be seen as an asymptotic improvement over Aichholzer's algorithm. As for the upper bound on t, ours is the first non-trivial bound on it to be known, however, we suspect that the real value should be closer to 4^n.

Now, no configuration of points is known having as many T-paths as triangulations. Hence, our T-path-based algorithm could potentially count triangulations *asymptotically* faster than enumeration algorithms. No similar result was known before, which makes ours worth mentioning. On the negative side, the bound for the running time of our algorithm is very precise, it depends on the *largest*[1] number of T-paths the algorithm encounters when run on P, and this number can get very large, sometimes at least $\Omega(4^n)$. Thus it is necessary to keep looking for better algorithms. In Chapter 5 we will see a result that goes in this direction.

4.1.2 The result on counting pseudo-triangulations

Pseudo-triangulations have been the subject of extensive research from the counting point of view, see [69, 15] and references therein. As of today it is not known whether, for *any* set of points, the number of pointed pseudo-triangulations is at least as large as its number of triangulations. Observe that if we remove the pointedness condition, the answer is trivially "yes".

In [6] the concept of *zig-zag path of a pseudo-triangulation* was introduced. This concept is for pseudo-triangulations what T-paths are for triangulations. For simplicity and consistency we will call such zig-zag paths simply *PT-paths*.

[1] Since T-paths are referenced by a line, different lines might generate different numbers of T-paths.

Definition 4.3 (PT-path). Given a planar set of points P, a pseudo-triangulation S of P, and a separating line l w.r.t. P, a PT-path of S w.r.t. l, denoted by $\mathrm{pt}(l, S)$, is defined as follows: (1) $\mathrm{pt}(l, S)$ is a chain of edges of S whose starting and ending edges are two edges of $\mathcal{CH}(P)$ intersected by l, and whose intersections with l are linearly ordered along l. (2) The area bounded by $\mathrm{pt}(l, S)$, between two consecutive intersections with l, and line l is an empty pseudo-triangle. (3) The reflex vertices of the empty pseudo-triangles of (2) are pointed in S. See to the right in Figure 4.2 for an example of a PT-path $\mathrm{pt}(l, S)$.

As for T-paths, an equivalent of Theorem 4.2 for PT-paths was proven in [6]:

Theorem 4.4 (O. Aichholzer, G. Rote, B. Speckmann, I. Streinu). *The PT-path* $\mathrm{pt}(l, S)$ *of a pseudo-triangulation* S *w.r.t. separating line* l *always exists and is* unique.

The previous theorem does not necessarily hold if we remove the pointedness condition, that is, a non-pointed pseudo-triangulation might contain more than one PT-path for the same reference line l. Nonetheless, for such cases one can still define a "canonical" PT-path.

Again, as for T-paths, divide-and-conquer algorithms that use PT-paths can be devised to count the elements of $\mathcal{F}_{\mathrm{PT}}(P)$, one such algorithm was already present in [6]. Those algorithms, as for T-paths, end up having running times of the sort $O\left(t^2\right)$, where $t = t(n)$ is the largest number of PT-paths of P, w.r.t. to some separating line l, that the algorithm encounters.

The result on pseudo-triangulation that we will prove is the following:

Theorem 4.5 (V. Alvarez, K. Bringmann, S. Ray). *Let* P *be a given set of* n *points on the plane. Then the exact value of* $|\mathcal{F}_{\mathrm{PT}}(P)|$ *can be computed in* $O\left(n^7 \cdot t\right)$ *time, and* $O(t)$ *space, where* t *is the largest number of PT-paths the algorithm encounters when run on* P.

Thus again, our result gives a significant improvement over known algorithms for counting pseudo-triangulations. This time, however, we are not able to show an upper bound on the largest number of PT-paths that can be constructed w.r.t. a given line.

The rest of the chapter is organized as follows: In § 4.2 we prove Theorem 4.3 and in § 4.3 we prove Theorem 4.5. We close the chapter in § 4.4 with discussions and conclusions.

4.2 Counting triangulations

Let T be a triangulation of P and let l be a separating line w.r.t. P. Without loss of generality we will assume that l is vertical. Let e be an edge of T properly intersecting

l. If e is not an edge of $\mathcal{CH}(P)$, we will say that e is *flippable* iff the union Q of the two triangles of T sharing e forms a convex quadrilateral. If Q is non-convex, or e is an edge of $\mathcal{CH}(P)$, we will simply say that e is *non-flippable*. Also, for Q, we will call the two vertices that are not vertices of e, the *opposite vertices* of e. Finally, we will say that e is *good* with respect to l iff e is flippable and its opposite vertices lie on different sides of l.

Now, let $p(l, T)$ be a T-path of T. The region between two consecutive edges $e = ab, e' = bd$ of $p(l, T)$, and delimited by l, defines a wedge $W = abd$ with apex at vertex b, see Figure 4.3.

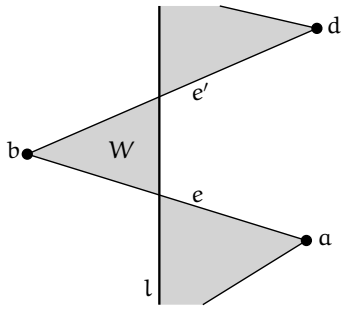

Figure 4.3 – Vertices a, b, d are three consecutive vertices of the shown T-path.

Observe that by part (3) of Definition 4.2, wedge W is empty of points of P, so we can define the set $\mathcal{W} = \mathcal{W}(p(l, T)) = \{W_1, W_2 \ldots, W_k\}$, as the set of all those empty wedges. Since we are going to use wedges throughout the whole section, whenever we have three consecutive vertices a, b, d of $p(l, T)$, we will use the shorthand abd to denote the corresponding element of $\mathcal{W}$ formed by the triple, in which the middle element is the apex. We now have the following observations:

Lemma 4.1. *Let T be a triangulation of P, let l be a vertical line, and let e be a good edge of T w.r.t. l. Then e is an edge of the unique T-path $p(l, T)$.*

Proof. Assume for the sake of contradiction that edge $e = pq$ of T is good but not an edge of $p(l, T)$, that is, edge e cannot be an edge of $\mathcal{CH}(P)$. Let $\mathcal{W}$ be the set of empty wedges of $p(l, T)$. Observe that every element W of $\mathcal{W}$ defines an interval on l, which is precisely where W intersects l, see Figure 4.4.

Note that every interior point of an interval on l defined by some element of $\mathcal{W}$ belongs only to that element of $\mathcal{W}$, that is, two intervals defined by two different elements of $\mathcal{W}$ have disjoint interiors. Denote by x the point of intersection between e and l. This point x cannot be the boundary point of any interval on l defined by some element of $\mathcal{W}$, otherwise there would be an edge $e' \neq e$ of $p(l, T)$ that crosses l at x, but that would

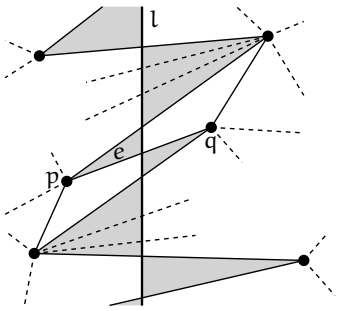

Figure 4.4 – Every empty wedge of $p(l, T)$ defines an interval on l where they intersect.

mean that e and e' intersect, which is clearly impossible since both edges belong to T, see Figure 4.5.

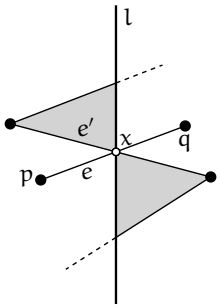

Figure 4.5 – The intersection between e and l cannot be the boundary of an interval on l defined by an empty wedge of $p(l, T)$.

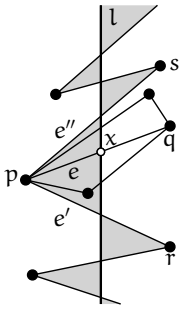

Figure 4.6 – Point x lies in the interior of the interval of l defined by the empty wedge with apex p. Since e is good w.r.t. l, the third vertex of one of the triangles of T that share e must lie inside W.

Thus x must belong to the interior of some interval on l defined by some element W of $\mathcal{W}$. It is also clear that the apex of W must be either p or q, otherwise, either p or q lies inside W, which is not possible since W is an empty wedge of $p(l, T)$. Let us assume without loss of generality that the apex of W is p, and that W is defined by the two consecutive edges $e' = rp$ and $e'' = ps$ of $p(l, T)$. Assume without loss of generality that p lies to the left of l, and thus r, q, s lie to the right. Note that x lies between the intersection points of e' and e'' with l, see Figure 4.6. Since e is good w.r.t. l, then the two triangles of T sharing e have their third vertices on different sides of l, which means that one of them necessarily lies inside W, which is again a contradiction since W is empty of vertices of T. Thus e must belong $p(l, T)$. ■

Observe that in general a T-path can also contain non-flippable edges.

Lemma 4.2. *Let* T *be a triangulation with vertex set* P*, and let* e *be a flippable edge of* T*. Then there exists a line* l *such that* e *is an edge of the* T*-path* $p(l, T)$*.*

Proof. Let $e = pq$ be a given flippable edge. Then e cannot be an edge of $\mathcal{CH}(P)$, thus, e is shared by two triangles of T, the third point of each triangle is r and s respectively. Let $e' = rs$ be the other diagonal of the convex polygon $prqs$, see Figure 4.7. Let l be the vertical line containing the point of intersection between e and e'. Then l makes e and e' good. ■

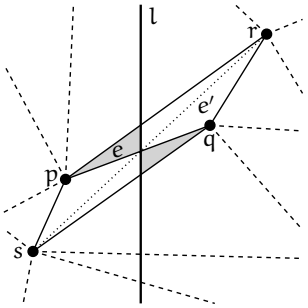

Figure 4.7 – e and e' are the two diagonals of the convex quadrilateral $prqs$. The line l containing their intersection makes both, e and e' good.

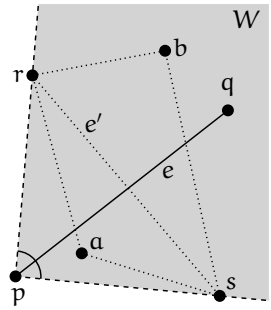

Figure 4.8 – Vertices a, b must be in the gray zone otherwise angle $\angle rps$ would not be maximum.

Lemma 4.3. *Let* T *be a triangulation with vertex set* P*. Then the set of all flippable edges of* T *is enough to characterize* T*.*

Proof. Let $F(T)$ be the set of all flippable edges of T. We have to prove that there cannot be another triangulation T' with vertex set P such that $T \neq T'$ but $F(T) = F(T')$.

Let us assume for the sake of contradiction that such triangulation T' exists. Define the set $NF(T) = E(T) \setminus F(T)$, which is the set of all non-flippable edges of T. Clearly, $NF(T) \neq NF(T')$, otherwise $T = T'$. That is, there must be at least one edge $e \in NF(T)$ that is properly intersected by edges of $NF(T')$; it cannot be intersected by edges of $F(T) = F(T')$, and both $NF(T), NF(T')$ cannot form a set of non-crossing edges since T and T' are sets of non-crossing edges of maximum cardinality, but $NF(T) \not\subseteq E(T')$ and $NF(T') \not\subseteq E(T)$.

Now let $e = pq$, and let $e' = rs$ be an edge of $NF(T')$ crossing e. Clearly, the edges of the quadrilateral $Q = prqs$ cannot be part of either T or T' because that would make e and e' flippable, see Figure 4.8. Assume that e' is the edge of $NF(T')$ crossing e that maximizes the angle $\angle rps$, such e' must exist. Given that all the edges of $\mathcal{CH}(P)$

are also shared by T and T' we have that e' must be shared by two triangles of T', so let a, b the third point of each triangle respectively, see Figure 4.8. Observe that it could happen that $p = a$, but then $b \neq q$, since quadrilateral Q makes e' flippable. Or vice-versa, $b = q$, but then $p \neq a$. Then a, b must be contained in the infinite wedge $W = rps$ with apex at p. Otherwise, say w.l.o.g. that a lies outside W. This means that another edge of triangle $\triangle rsa$, other than e', intersects e properly. Say edge ra. But then angle $\angle rpa > \angle rps$, which is a contradiction since $\angle rps$ was chosen to be maximum among all the edges of $NF(T')$ crossing e. Note however that if a, b are contained in W, then the quadrilateral $rasb$ is convex, which means that e' is flippable in T', which is a contradiction since we assume that $e' \in NF(T')$. Hence such an edge $e' \in NF(T')$ crossing e cannot exist, which means that $NF(T') = NF(T)$, since e was an edge of $NF(T)$, and thus we arrive at $T = T'$. ■

Lemma 4.4. *Let T be a triangulation with vertex set P, and let l, l' be two vertical lines such that $l \neq l'$, and the vertical slab between l and l' is empty of points of P. Then $p(l, T) = p(l', T)$.*

Proof. Let us assume without loss of generality that l' lies to the left of l. Since the vertical slab between l' and l is empty of points of P, observe that there is a bijection between the set $\mathcal{W}(p(l, T))$, the empty wedges of $p(l, T)$, and the set $\mathcal{W}(p(l', T))$, see Figure 4.9.

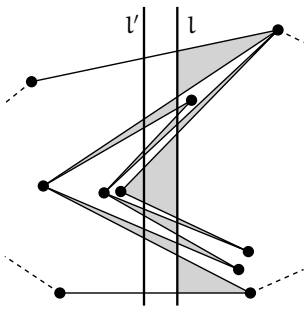

Figure 4.9 – T-path $p(l, T)$ shown, along its empty wedges. Every wedge is also empty w.r.t. l'.

Thus $p(l, T)$ and $p(l', T)$ are both T-paths, by definition, of T w.r.t. l' and l respectively, but every T-path of T w.r.t. some line is unique, so there is no other option but $p(l, T) = p(l', T)$. ■

Now let us assume that P is sorted from left to right, *i.e.*, from smallest x-coordinate to the largest. We can assume that by a suitable rotation we do not have any ties in the x-coordinate, so $P = \{p_1, p_2, \ldots, p_n\}$.

Let $\mathcal{L} = \{l_1, \dots l_{n-1}\}$ be a set of vertical lines such that point $p_i \in P$ lies in the vertical slab between l_{i-1} and l_i, with $2 \leqslant i \leqslant n-1$. Point p_1, the leftmost, lies in the unbounded vertical slab to the left of l_1, and p_n, the rightmost, lies in the unbounded vertical slab to the right of l_{n-1}. For a triangulation T of P let $\mathcal{P}(T) = \{p(l_i, T) \mid l_i \in \mathcal{L}\}$. We now have the following result:

Theorem 4.6. *Let T be a triangulation with vertex set P. Then $\mathcal{P}(T)$ is enough to characterize T.*

Proof. We have to prove that there cannot be another triangulation T' with vertex set P such that $T' \neq T$, but $\mathcal{P}(T) = \mathcal{P}(T')$. However, by Lemma 4.3 we know that the set of flippable edges of a triangulation characterizes it, hence it is enough to prove that *every* flippable edge of T is an edge of some T-path in $\mathcal{P}(T)$.

Let us assume for the sake of contradiction that there is an edge e of T that is flippable but that is not an edge of any T-path in $\mathcal{P}(T)$. By Lemma 4.2 we know that there exists one vertical line l such that e is an edge of the T-path $p(l, T)$. Note that such a line l is parallel to every line in $\mathcal{L}$, and that one endpoint of e lies to the left of l and the other to the right, so l must lie inside the vertical slab between to consecutive lines of $\mathcal{L}$, or to the left of $l_1 \in \mathcal{L}$, or to the right of $l_{n-1} \in \mathcal{L}$, let us assume without loss of generality that l lies in the vertical slab between l_i and l_{i+1}, with $1 \leqslant i \leqslant n-2$. Observe however that such a slab contains exactly one point of P, thus it must happen that either, the vertical slab between l_i and l is empty of points of P, or the vertical slab between l and l_{i+1} is empty of points of P, say the former without loss of generality. Nevertheless, by Lemma 4.4, we know that $p(l, T) = p(l_i, T)$, so $p(l, T) \in \mathcal{P}(T)$, which is a contradiction since e was a flippable edge of T that was not an element of $\mathcal{P}(T)$. Thus, such an edge e cannot exist, and there is no other option but $T = T'$ since they share the same set of flippable edges. ■

Therefore every triangulation T having P as vertex set has a unique set $\mathcal{P}(T)$ of T-paths, and thus the number of triangulations $|\mathcal{F}_T(P)|$ is just the number of different sets of T-paths $\mathcal{P}(T)$ that we can find on P. Let $\Pi(l, P) = \{p(l, T) \mid T \text{ is a triangulation of } P\}$ be the set of all T-paths of P w.r.t. line l. Note that while the set of lines $\mathcal{L}$ stays fixed, there will be in general more than one T-path that can be formed per line, thus a tuple $\{\pi_1, \dots, \pi_{n-1}\}$ of T-paths of P, with $\pi_i \in \Pi(l_i, P)$, defines a triangulation if and only if all those T-paths are pairwise non-crossing. We will say that such a pairwise non-crossing set is *compatible*. It is easy to show that, in order to verify if such a set is compatible, it suffices to check that two consecutive T-paths $\pi \in \Pi(l_i, P)$ and $\pi' \in \Pi(l_{i+1}, P)$ are non-crossing, for $1 \leqslant i \leqslant n-2$.

Note that there might be triangulations sharing some T-paths, for example, if P is in convex position, its number of triangulations is $O(4^n)$, while its number of T-paths is

$O(2^n)$, so we obtain on average $O(2^n)$ triangulations per T-path. This motivates the following definition:

$$\mathcal{T}(\pi_j) = \{\{\pi_1, \ldots, \pi_{j-1}\} \mid \{\pi_1, \ldots, \pi_{j-1}, \pi_j\} \text{ is compatible and } \pi_i \in \Pi(l_i, P)\}.$$

We need two more definitions in order to describe our algorithm. For each $\pi' \in \Pi(l_{i+1}, P)$ we define $\lambda(\pi') = \{\pi \in \Pi(l_i, P) \mid \pi \text{ is compatible with } \pi'\}$. Similarly we define $\mu(\pi) = \{\pi' \in \Pi(l_{i+1}, P) \mid \pi' \text{ is compatible with } \pi\}$ for each $\pi \in \Pi(l_i, P)$. Now we are ready to describe our algorithm.

4.2.1 The sweep line algorithm

We consider sweeping a vertical line from left to right, the *event points* being the vertical lines in the set $\mathcal{L}$ as defined before. At any event point l_i we maintain $\Pi(l_i, P)$, and for each $\pi \in \Pi(l_i, P)$ we store $|\mathcal{T}(\pi)|$. At $i = 1$ we clearly have $|\Pi(l_1, P)| = 1$, and for this particular $\pi \in \Pi(l_1, P)$ we have $|\mathcal{T}(\pi)| = 1$. We will show that each $\pi' \in \Pi(l_{i+1}, P)$ can be obtained from each $\pi \in \Pi(l_i, P)$ compatible with π'[II] by doing *local changes*, which will be defined later on, for the time being the important thing to know is that the number of possible local changes for a T-path is $O(n^2)$. Hence, if we go through each $\pi \in \Pi(l_i, P)$ and try all possible local changes for π, we will obtain $\Pi(l_{i+1}, P)$. Moreover, for each $\pi' \in \Pi(l_{i+1}, P)$ we also get the set $\lambda(\pi')$. Observe that $|\mathcal{T}(\pi')|$ is given by $\sum_{\pi \in \lambda(\pi')} |\mathcal{T}(\pi)|$. Thus we are able to compute $\Pi(l_{i+1}, P)$ as well as $|\mathcal{T}(\pi')|$ for each $\pi' \in \Pi(l_{i+1}, P)$. All this takes time $O(n^2 \cdot t_i)$, where $t_j = |\Pi(l_j, P)|$, since there are $O(n^2)$ local changes to try for each $\pi \in \Pi(l_i, P)$, and as we will see later, the time taken per local change is constant. The overall running time of the algorithm is therefore $\sum_{l_j \in \mathcal{L}} O(n^2 \cdot t_j) \leqslant O(n^3 \cdot t)$, where $t = \max\{t_j\}$. At the end, the number we are looking for is precisely $|\mathcal{F}_T(P)| = |\mathcal{T}(\pi)|$, where π is the unique T-path of $\Pi(l_{n-1}, P)$.

Our main task now is to explain the local changes and to prove that there are indeed $O(n^2)$. We first need the following intermediate result:

Lemma 4.5. *At times $l = l_i$ and $l = l_{i+1}$, point $p = p_{i+1}$ has degree zero, one, or two in every T-path $\pi \in \Pi(l_i, P)$, as well as in every T-path $\pi' \in \Pi(l_{i+1}, P)$. However, if π and π' do not cross, then p cannot simultaneously have degree zero in both T-paths, that is, p must be a vertex of at least one T-path.*

Proof. Let us look at the case when $l = l_i$, the other case, $l = l_{i+1}$ is just symmetric. If p is not a vertex of π, then the degree of p is zero. If p is a vertex of π, then there

[II] Again, by compatibility we mean non-crossing.

are two cases depending on whether p is a vertex of $\mathcal{CH}(P)$ or not. Since both cases are very similar we will prove only the latter.

Since p lies inside $\mathcal{CH}(P)$ we know that p is an internal vertex of π, *i.e.*, the degree of p in π is at least two. To verify that it is at most two let us assume that its degree is at least four, it must be even. Let b be the first neighbor of p in π, when visiting p while traversing π from the first vertex to the last. Similarly, let c be the last neighbor of p in π in the same traversing order, see Figure 4.10. Since the degree of p is at least four, there must be other two vertices b', c' between b and c. Observe that p lies to the right of l, and b, b', c, c' to the left, so there must be at least one vertex $x \neq p$ of π connecting b' and c', however, x should lie inside the vertical slab between l and l_{i+1}, which is empty of points of P except for p, see Figure 4.11. Thus x cannot exist, which implies that b', c' cannot exist either. Hence, the degree of p in π is at most two, which is what we wanted to prove.

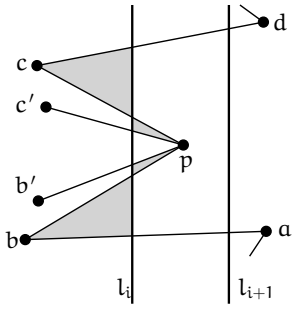

Figure 4.10 – T-path $\pi \in \Pi(l_i, P)$ where p has degree at least four shown.

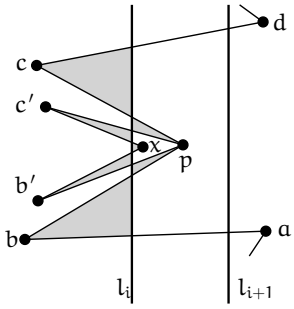

Figure 4.11 – Vertex x of π cannot exist because p is the only point in that vertical slab.

It remains to prove that p cannot have degree zero in both T-paths, π and π', if they do not cross. To see this, note that if neither π nor π' has p as a vertex, then clearly p cannot be on $\mathcal{CH}(P)$, so p must lie in the interior of $\mathcal{CH}(P)$, and thus it also lies inside the triangles $\triangle abd$, and $\triangle a'b'd'$, where a, b, d and a', b', d' are consecutive vertices of π and π' respectively, see Figures 4.12 and 4.13. Note however that this case can only happen if either $\triangle abd$ and $\triangle a'b'd'$ intersect, or if one lies entirely inside the other, since both triangles contain p in their interior. In the first case we have obviously an intersection between π and π', which is a contradiction. In the second case, assume without loss of generality that $\triangle abd$ lies inside $\triangle a'b'd'$. But then observe that since a, d and b' lie on the same side of l_{i+1}, then the wedge $a'b'd'$ of π' is not empty, which is clearly not possible since π' is a T-path and edges $a'b'$ and $b'd'$ are consecutive in π', see Figure 4.13. Thus, Lemma 4.5 follows. ■

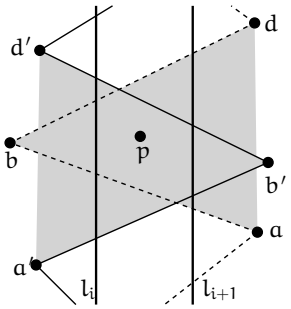

Figure 4.12 – π is shown in solid lines, and π' in dashed lines.

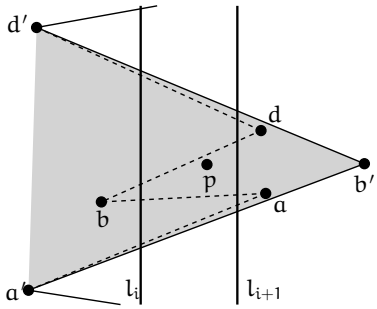

Figure 4.13 – If $\triangle abd$ lies inside $\triangle a'b'd'$, then the wedge $a'b'd'$ with apex b' and delimited by l_{i+1} is not empty.

We are now ready to explain the local changes carefully: From Lemma 4.4 we know that $p(l, T) = p(l', T)$ for a triangulation T of P as long as the vertical slab between l and l' is empty of points of P. This in turn implies that $\Pi(l, P) = \Pi(l', P)$. Now assume that $l' = l_i$ and $l = l_{i+1}$, that is, the vertical slab between l and l' is no longer empty, but contains point $p = p_{i+1}$. It is clear that during the continuous movement from l_i to l_{i+1} the only ways a T-path can change, are the ones involving p in the following two senses: If p is not a vertex of the current T-path $\pi \in \Pi(l_i, P)$, then the only empty wedge of π that cannot be made an empty wedge of a T-path $\pi' \in \Pi(l_{i+1}, P)$ is the one that during the sweeping process starts containing p, see Figure 4.14. If on the other hand, p is a vertex of π, then its neighbors in π lie to the left of l_i, since p lies to the right, see Figure 4.15. But then p along with its neighbors lie to the left of l_{i+1}, so those adjacencies cannot be part of a T-path w.r.t. l_{i+1}. Thus we will obtain $\mu(\pi)$, for every T-path $\pi \in \Pi(l_i, P)$, by locally changing π around p. We will have two cases to consider depending on whether p appears as a vertex of the current T-path $\pi \in \Pi(l_i, P)$ we are considering, or not. We will study each case in turn, however, there is a case analysis that one has to do, so in order to avoid going through all cases, we will describe the general setting from which all the cases can be obtained. Let again $\pi \in \Pi(l_i, P)$:

(1) Assume that p appears as a vertex of π, and let us first consider the case when p lies in the interior of $\mathcal{CH}(P)$. By Lemma 4.5 point p must have degree exactly two in π.

Now take vertices a, b, c, d of π as displayed in Figure 4.16. Look for all pairs of points $b', c' \in P$ such that the substitution of the pattern (a, b, p, c, d) in π to (a, b, b', p, c', c, d) results in a T-path w.r.t. l_{i+1}, see Figure 4.16.

Observe that as particular cases we could have $b' = c' = a = d$, which would result in the substitution $(a, b, p, c, d) \to (a)$, or we could have $b' = c' = a$, $a \neq d$, which would result in $(a, b, p, c, d) \to (a, c, d)$. Since there are many cases,

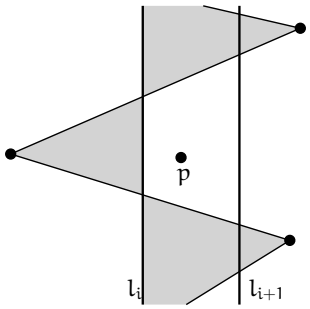

Figure 4.14 – Sweeping from l_i to l_{i+1} results in a wedge containing p.

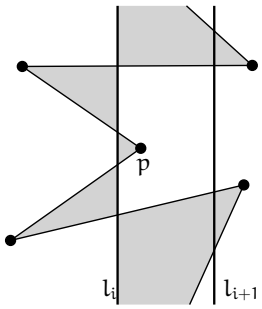

Figure 4.15 – Sweeping from l_i to l_{i+1} results in the adjacencies of p being on the same side of l_{i+1}.

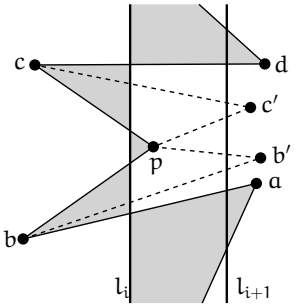

Figure 4.16 – Substitution $(a, b, p, c, d) \to (a, b, b', p, c', c, d)$ is only one of the possibilities.

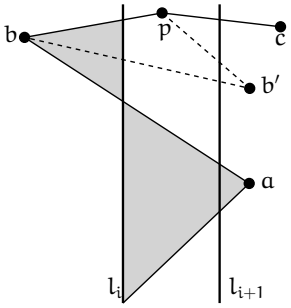

Figure 4.17 – Substitution $(a, b, p) \to (a, b, b', p, c)$.

we would have to exhaust all choices for b', c', however, they all occur inside the same region.

If $p \in \mathcal{CH}(P)$, then p could be the very first vertex of π, or the very last, or the second, or second-to-the-last. Let us consider when p is the last, it is symmetric to the case when p is the first. Let the last three vertices of π be a, b, p in that order, so $b \in \mathcal{CH}(P)$ as well, and bp is intersected by l_i. We are looking in general for the substitution $(a, b, p) \to (a, b, b', p, c)$, where $c \in \mathcal{CH}(P)$ is the other neighbor of p on $\mathcal{CH}(P)$. Observe that pc is intersected by l_{i+1}, see Figure 4.17. We could for example have $b' = c$ or $b' = a$ as particular cases, among others.

(2) Now assume p does not appear as a vertex of π. Then p cannot be a vertex of $\mathcal{CH}(P)$ either, as otherwise one of the edges of $\mathcal{CH}(P)$ having p as a vertex would intersect l_i, and thus p would necessarily appear in π by definition. Thus π must look locally as in Figure 4.18, that is, the point p must be contained inside the triangle $\triangle abd$, where a, b, d are consecutive on π, point b lies on one side of l_i, and a, d on the other side. Thus observe that the adjacency bp is forced in

any triangulation containing π, since p is the only point of P contained in the vertical slab between l_i and l_{i+1}. The reader will be able to verify that this case is a particular case of (1) in which $b = c$, and we could have, for example, the substitutions $(a, b = c, d) \to (a, p, d)$, or $(a, b = c, d) \to (a, b, b', p, c', b, d)$, among others, see Figure 4.17.

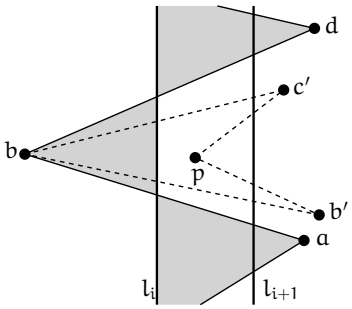

Figure 4.18 – Substitution $(a, b, d) \to (a, b, b', p, c', b, d)$.

Note that the substitutions can be done in reverse order, that is, imagine that we go back in time, from time $l = l_{i+1}$ to time $l = l_i$, so we would be sweeping the plane from right to left, and therefore the pattern (b, b', p, c', c) of some $\pi' \in \Pi(l_{i+1}, P)$ could become pattern (b, p, c) of some $\pi \in \Pi(l_i, P)$, upon proper relabeling of points, see Figures 4.16, 4.17 and 4.18. So π' is obtained from π in one direction, and π is obtained from π' in the opposite direction, this relation will be denoted by $\pi \leftrightarrow \pi'$. We have finally the following result:

Lemma 4.6. *Given $\Pi(l_i, P)$, every T-path of $\Pi(l_{i+1}, P)$ is produced by the local changes just explained. Moreover, for each $\pi \in \Pi(l_i, P)$, the cardinality of $\mu(\pi)$ is $O(n^2)$, and we can correctly compute $\lambda(\pi')$ for each $\pi' \in \Pi(l_{i+1}, P)$ in time $O(n^2 \cdot t_i)$.*

Proof. Let again $p = p_{i+1} \in P$. For the first part let $\pi' \in \Pi(l_{i+1}, P)$. We will prove that π' produces at least one T-path $\pi \in \Pi(l_i, P)$. The result will then follow by the relation $\pi \leftrightarrow \pi'$ explained before. For the second part we have to show that $|\mu(\pi)| = O(n^2)$ for each $\pi \in \Pi(l_i, P)$, and that we are able to correctly compute $\lambda(\pi')$ for each $\pi' \in \Pi(l_{i+1}, P)$ in time $O(n^2 \cdot t_i)$. That is, we will prove that if $\pi \not\leftrightarrow \pi'$ then both T-paths cross, and thus $\pi \notin \lambda(\pi')$. For both parts we have two cases depending on whether p is a vertex of π' or not, but for simplicity we will only consider the case when p is not a vertex of π', the other case in both parts follows using similar arguments.

Let W be the empty wedge of π' that cannot be extended to an empty wedge W' of π due to p. Thus p lies inside the triangle $\triangle abd$, where a, b, d are consecutive vertices, see Figure 4.19. Let ap, pd be two new adjacencies. Observe that a, d lie to the left of

l_i, and p, b lies to the right. If the substitution $(a, b, d) \to (a, p, d)$ results in a T-path of $\Pi(l_i, P)$, we are done, if not, then the triangle $\triangle bap$, or the triangle $\triangle pdb$ is not empty, probably even both. Let us assume without loss of generality that the former is the one that is not empty, and that this is the only one. If both triangles contain points of P we can proceed in the same way on both of them. Call this non-empty triangle $\triangle'$, and observe that there is at least one point $c' \in P$ contained in $\triangle'$. Choose it and create the adjacencies $bc', c'p$. Now do the substitution $(a, b, d) \to (a, b, c', p, d)$, and again test if the new path is an element of $\Pi(l_i, P)$. If yes, we are done, if not, set $\triangle' = bc'p$, and thus, there must be again some point of P inside $\triangle'$. Choose one of those points, label it with c', and repeat. Observe that every new point we take lies to the left of l_i. Since P is finite, we will eventually arrive at $\triangle'$ being empty, and at that point, we would have created an element of $\Pi(l_i, P)$, see Figure 4.20.

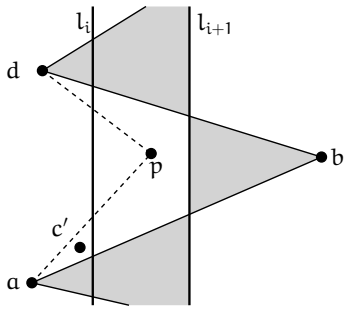

Figure 4.19 – T-path π' is shown in solid.

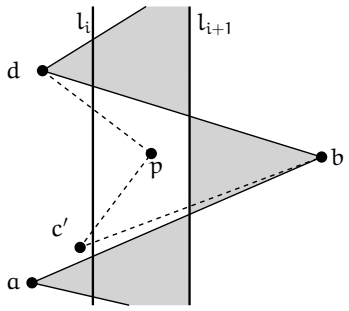

Figure 4.20 – T-path π' being extended to a T-path $\pi \in \Pi(l_i, P)$.

For the second part, by the way the local changes are made, it is clear that from a T-path $\pi \in \Pi(l_i, P)$ we cannot obtain more than $O(n^2)$ T-paths of $\Pi(l_{i+1}, P)$, since when trying local changes of π around p, at most every pair of points of P will be tested, and thus every such a pair can produce at most one T-path of $\Pi(l_{i+1}, P)$. We now have to prove that if $\pi \not\leftrightarrow \pi'$ then π and π' cross. Remember that we are still assuming that p is not a vertex of π', thus p is still inside triangle $\triangle abd$, where a, b, d are three consecutive vertices of π'. Let us assume for the sake of contradiction that $\pi \not\leftrightarrow \pi'$, but $\pi \in \lambda(\pi')$, *i.e.*, those two paths are non-crossing. Since $\pi \in \lambda(\pi')$ there must be at least one triangulation of P containing both T-paths. Let T be one of those triangulations, and observe that in T, vertex p must have at least two adjacencies to the left of l_i, since the degree of p in π' is zero. Among all these adjacencies keep just the first and the last in the radial order around p in clockwise order. Let b', c' be those two neighbors of p respectively, see Figure 4.21. Clearly b' and c' must be adjacent to b, but then the substitution $(a, b, d) \to (a, b, b', p, c', b, d)$ creates a T-path $\pi'' \in \Pi(l_i, P)$, that is, $\pi'' \leftrightarrow \pi'$, and thus we have that $\pi \neq \pi''$ since $\pi \not\leftrightarrow \pi'$. But π'' is also a T-path of T

w.r.t. l_i, which is a contradiction since the T-path of a triangulation w.r.t. a given line is *unique*, hence such π' cannot exist.

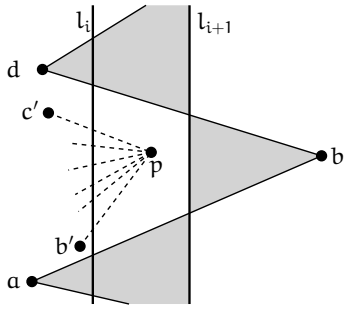

Figure 4.21 – In any triangulation of P containing π', vertex p must have at least two adjacencies to the left of l_i.

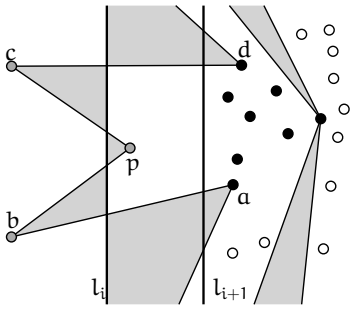

Figure 4.22 – All possibilities for b', c' are shown as black points. The white points are visible from neither b nor c.

It remains to prove that $\lambda(\pi')$ can be computed in time $O(n^2 \cdot t_i)$ for each $\pi' \in \Pi(l_{i+1}, P)$, where $t_i = |\Pi(l_i, P)|$. From the discussion above we obtain that $\pi \in \lambda(\pi')$ if and only if $\pi \leftrightarrow \pi'$. The relation $\pi \leftrightarrow \pi'$ is obtained by guessing pairs of points b', c', and checking if the new adjacencies, attached to π, produce π'. For example, let us assume we want to obtain the possible substitutions for the pattern (a, b, p, c, d), with $p = p_{i+1}$, like in Figure 4.22. We just have to look for b', c' among all the points of P that are visible from b or c, having the edges of π as obstacles, see Figure 4.22. All these points can be obtained in $O(n^2)$ time, since the number of edges of π is $O(n)$. Once we obtain this list of candidates, one list B for b and another list C for c, we try every possible pair b', c' such that $b' \in B$, and $c' \in C$, for adjacencies that would create π', for example, we could try adjacencies $bb', b'p, pc', c'c$ to obtain the substitution $(a, b, p, c, d) \to (a, b, b', p, c', c, d)$, but if $c' = d$ occurs, then we would have to try substitution $(a, b, p, c, d) \to (a, b, b', p, d)$, and so on depending on the particular configuration. If we pre-process P in such a way that we can answer in constant time if a given triangle with vertices in P is empty or not, we can also test the correctness of the adjacencies in constant time per pair b', c'. Thus we spend overall $O(n^2)$ time per path π of $\Pi(l_i, P)$. If we have that $\pi \leftrightarrow \pi'$, then we also have that $\pi \in \lambda(\pi')$, and thus after $O(n^2 \cdot t_i)$ we have constructed $\lambda(\pi')$, for every $\pi' \in \Pi(l_{i+1}, P)$, where $t_i = |\Pi(l_i, P)|$. This completes the proof. ■

The above discussion implies the algorithmic part of Theorem 4.3. The next subsection addresses the second part of the same theorem, *i.e.*, a rough upper bound, depending only on n, for the running time of the algorithm just presented will be given.

4.2.2 On the number of triangulation paths

It is known that if P is in convex position, then the largest number of T-paths that we can find w.r.t. some line is $O(2^n)$, see [2]. However, there could be configurations for which this number is much larger. In [32] a set P is shown for which we can find $\Omega(4^{n-\Theta(\log(n))})$ T-paths w.r.t. to some line. This number is essentially 4^n, thus we can see that the number of T-paths that one needs to consider is also large. Up to now there have been no results about the largest number of T-paths, over all sets of n points on the plane, and over all possible lines we can define T-paths on. The main result presented here is the following:

Theorem 4.7. *The largest number of T-paths, w.r.t. a line, of a set of n points P on the plane is at most $O(9^n)$.*

Before the actual proof, let us first explain how we are going to count T-paths. Let P be a set of n points whose elements are labeled with the integers from 1 to n, and let π be a T-path of P w.r.t. some given line l. Without loss of generality assume that π starts at the edge of $\mathcal{CH}(P)$ with the lowest intersection with l, and thus it ends at the edge of $\mathcal{CH}(P)$ with the highest intersection with l. Observe that given l, the starting and ending edges of any T-path w.r.t. l are always the same two edges of $\mathcal{CH}(P)$. Without loss of generality we will assume that π starts to the left of l, unless it is otherwise explicitly stated. If π starts to the right of l then we would have a symmetric conversation.

Now orient the edges of π as traversing it from the starting edge to the ending edge. The starting edge, by assumption, crosses l from left to right, the second from right to left, the third from left to right again, and so on until we arrive at the ending edge. Observe that the edges of π appear sorted bottom-up on l as they intersect l, so the starting edge has the lowest intersection with l, the second edge has the second lowest intersection with l, and so on. Thus the starting vertex of π and the edges of π that cross l from left to right are enough to characterize π. There is no other way one can complete adjacencies, since in-between two edges e, e'' crossing l from left to right, there must be an edge e' crossing from right to left and interconnecting e and e'', and vice-versa, see Figure 4.23. The starting vertex of π tells us if the starting edge crosses l from left to right or from right to left. Now let $e = p_i p_j$ be an edge of π that crosses l from left to right. Let us mark the intersection of e and l with the pair (i, j). Doing this for every edge of π that crosses from left to right we obtain a sequence N of pairs of integers on l, which along with the first vertex of π can be considered as the "signature" of π, since we know at each of those intersection points which edge of π crosses l, and in which direction. There is the particular case when π also ends to the left of π, and thus its last edge crosses l from right to left, and under our labeling scheme, the last vertex of π might not appear in any pair of integers on l, however, given l, the last edge of π is fixed, thus there is no confusion as how to complete π see Figure 4.23. Now, observe

that the sequence N of pairs of integers along l can be partitioned into the sequence N^- of vertices of π lying to the left of l, and the sequence N^+ of vertices of π lying to the right. Both sequences N^- and N^+ can be seen as sequences of integers that are sorted w.r.t. the order they appear on l bottom-up. The way we are going to upper-bound the number of T-paths of P w.r.t. l is by upper-bounding the number of different sequences that represent N^-. The same bound will obviously hold for the number of different sequences that represent N^+. The final bound will come out essentially from combining the two bounds obtained.

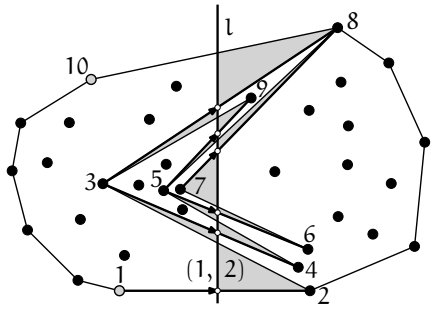

Figure 4.23 – A T-path π. The first and last vertices are shown in gray. The edges of π crossing l from left to right are shown with arrows, and the intersection point is shown as a white dot. The integer sequence N^- for π is 1,3,5,7,5,3.

Proof of Theorem 4.7. To create a sequence of integers that represent N^- we just need the elements of P that lie to the left of l. Let us denote this subset of points by $P^- \subset P$. Let $P_k \subseteq P^-$ be a subset of P^- of k elements. Imagine that the sequence N^- will be obtained using only elements of P_k, but *every* element of P_k must appear in N^- at least once. Let us assume without loss of generality that 1 is the leftmost point of P_k. Since 1 must appear in N^-, it means that there must be at least one straight-line segment s that connects 1 with l, this segment can be thought of as the left part of an edge of a T-path where 1 appears. Moreover, assume that s is the segment that creates the last entry of 1 in N^-, that is, point 1 is not connected to l at a higher point than the one that s defines. Thus s divides the problem into two sub-problems, since we want to keep everything non-crossing. Let P_k^- be the set of points of P_k above segment s, and let P_k^+ be the set of points of P_k below s but also including 1. There are k possibilities for P_k^-, since we can rotate s around 1 clockwise to make the cardinality of P_k^- vary from 0 to $k-1$, and thus the cardinality of P_k^+ varies from k to 1. Since we are assuming that s is the segment that connects point 1 for the last time to l, then point 1 does not form part of the sub-problem defined by P_k^-, thus this sub-problem is totally independent and we can recurse directly on it. However, point 1 does play a role in the sub-problem defined by P_k^+. If $f(k) = f_k$ represents the total number of different possibilities for N^- when k

points are involved, then we get the following recurrence for f_k:

$$f_k = g_k + \sum_{i=1}^{k-1} f_i \cdot g_{k-i}$$

where g_j represents the sub-problem defined by P_k^+, for every $1 \leqslant j \leqslant k$. Note that for $j = k$ we obtain that P_k^- is empty, and thus $|P_k^+| = k$, which is represented by the term g_k of f_k. Observe that in the case $j = k$, the sub-problem defined by P_k^+ is of the same size as the original problem, however, it has a slightly different structure, since in P_k^+ we know that point 1 is already connected to l, so the immediate lower connection of 1 to l, if any, cannot be consecutive: This would mean that there are two consecutive edges e, e'', of some T-path, crossing l from left to right, and sharing vertex 1 as endpoint, but between e, e'' there must be exactly one edge e' of the same T-path that crosses l from right to left, see Figure 4.24. If we assume that e intersects l below e'', then e' intersects l in-between, and connects the right endpoint of e with the left endpoint of e'', thus $e = e'$, but in a T-path every edge is used exactly once, hence there cannot be two consecutive appearances of an integer in N^-. The summation term of f_k accounts for the other $k-1$ possibilities for P_k^- and P_k^+.

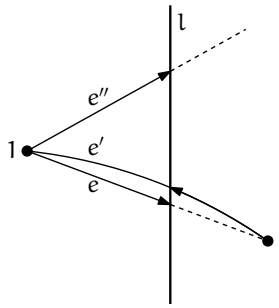

Figure 4.24 – Edges e, e'' are consecutive edges, of a T-path, that cross l from left to right and share vertex 1.

The recurrence for g_k is very similar; once we enter sub-problem P_k^+ we just have to take care of not connecting point 1 to l consecutively, so we have that:

$$g_k = h_k + f_{k-1} + \sum_{i=1}^{k-1} f_i \cdot g_{k-i}$$

where the term f_{k-1} means that point 1 is not used in P_k^+. If on the other hand, point 1 is used, then the summation will again account for all the possibilities. The term h_k is technical, and its definition is: $h_k = 1 \Leftrightarrow k = 1$, and 0 otherwise. With it we can safely define our boundary condition $f_0 = g_0 = h_0 = 0$, and we obtain $f_1 = g_1 = h_1 = 1$, which makes the recursion safe.

We are now interested in the asymptotic behavior of f. We will obtain it by using ordinary generating functions. We will not explain every single step in detail since we will be using standard techniques. The interested reader is referred to [83, 43] for the common techniques to obtain generating functions from recurrences.

Introducing the ordinary generating functions $F(x) = \sum_{k=0}^{\infty} f_k \cdot x^k$, $G(x) = \sum_{k=0}^{\infty} g_k \cdot x^k$, $H(x) = \sum_{k=0}^{\infty} h_k \cdot x^k = x$, we obtain for f_k, g_k the following:

$$F(x) = G(x) + F(x) \cdot G(x)$$
$$G(x) = H(x) + x \cdot F(x) + F(x) \cdot G(x)$$

We can now solve this system of equations in unknowns $F(x), G(x)$ to obtain two possible solutions for $F(x)$:

$$F_1 = F(x) = \frac{1 - \sqrt{1-8x}}{3 + \sqrt{1-8x}} \quad \text{and} \quad G_1 = G(x) = \frac{1 - \sqrt{1-8x}}{4}$$
$$F_2 = F(x) = \frac{-1 - \sqrt{1-8x}}{\sqrt{1-8x} - 3} \quad \text{and} \quad G_2 = G(x) = \frac{1 + \sqrt{1-8x}}{4}$$

However, we know that $F(0)$ must be 0, and this condition is only met by F_1, so $\frac{1-\sqrt{1-8x}}{3+\sqrt{1-8x}}$ is the generating function of our sequence f, *i.e.*, the coefficients of the Taylor expansion of F_1 around 0 are precisely the terms $f_0 = 0, f_1 = 1, f_2 = 3, f_3 = 13, f_4 = 67, f_5 = 381, f_6 = 2307\ldots$, which turned out to be known as sequence A064062 of "The On-Line Encyclopedia of Integer Sequences", but with term $f_0 = 1$, which makes no difference for the asymptotics of f, see [60]. The generating function of A064062 is $F_A = \frac{1}{1-xC(2x)}$, where $C(y) = \frac{1-\sqrt{1-4y}}{2y}$ is the generating function of the Catalan numbers, see [60] and references therein. It is now easy to verify that $F_A = F_1 + 1$, since F_A and F_1 differ only at $f_0 = 1$.

It is known that the i-th term f_i of F_A, for sufficiently large i, grows roughly as $\frac{8^i}{36i\sqrt{\pi \cdot i}} < 8^i$, see [60] and Theorem 3 of [19].

Thus the number of different possibilities for N^- that we can obtain from a set of cardinality k is upper-bounded by 8^k. It remains to consider every possible set $P_k \subseteq P^-$. If $|P^-| = a$, then the absolute number t^- we are looking for is upper-bounded by $\sum_{i=0}^{a} \binom{a}{i} 8^i = 9^a$. The same bound holds for the number t^+ of different sequences that represent N^+. If we partition the original set P into P^- of cardinality a, and P^+ of cardinality b, such that $a + b = n$, then the number of ways we can create T-paths of P w.r.t. l that start to the left of l is upper-bounded by $t^- \cdot t^+ = 9^a \cdot 9^b = 9^n$. The same bound holds for T-paths that start to the right of l, thus obtaining $O(9^n)$ overall possibilities. The theorem follows. ■

This concludes the proof of Theorem 4.3.

4.3 Counting pseudo-triangulations

The main idea behind our algorithm for counting pseudo-triangulations is to mimic with PT-paths what we did with T-paths for counting triangulations. Thus, here we will have equivalent results to the ones we proved in § 4.2. We will first explain how a PT-path $\mathrm{pt}(l, S)$ of a pseudo-triangulation S, with respect to line l, can be constructed, but in order to do so, we need to define some terms first.

Let l be a separating line, and let S be a pseudo-triangulation of P. Let us denote by E_l the set of edges of S that are intersected by l. Let $e \in E_l$ and denote by $\overline{e}$ and $\underline{e}$ the edges of E_l right above and below e respectively. We will say that $e \in E_l$ of S is *good*[III] w.r.t. l iff the intersections of the supporting line of e with the supporting lines of $\overline{e}$ and $\underline{e}$ lie on different sides of l, or if e is an edge of $\mathcal{CH}(P)$.

Let us now explain how a PT-path $\mathrm{pt}(l, S)$ of a pseudo-triangulation S, and with respect to line l, can be constructed. The following method was originally described in [6]: Remove from S all edges of E_l that are not good. This leaves a plane graph S^* of P. Let e and e' be two consecutive good edges w.r.t. l, and connect them using the common face f of S^* that they are part of according to the following rule: If the supporting lines of e and e' intersect to the left of l, then we use the edges of f that lie to the left. Otherwise we use the edges of f that lie to the right of l, see Figure 4.25.

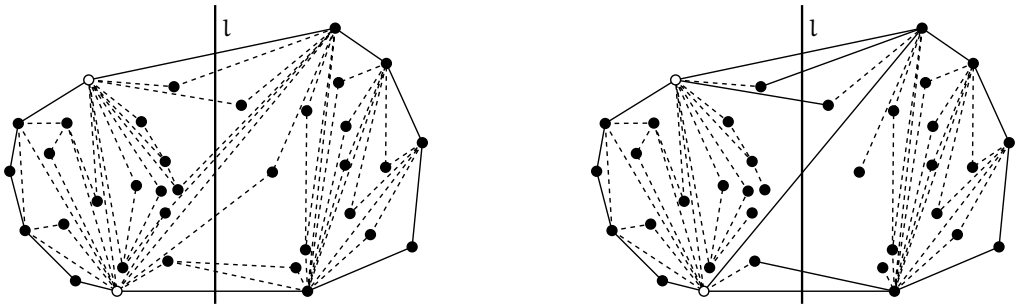

Figure 4.25 – To the left a pseudo-triangulation S. To the right we have the plane graph S^* obtained from S by removing all *non-good* edges of E_l. Joining two consecutive good edges of E_l by the rules described before results in the PT-path shown in Figure 4.2 on page 56.

Observe that the polygonal chain of edges created by the method described above always exists. In [6] it was proven that it fulfills the properties of a PT-path, see Definition 4.3 on page 58. Thus, by Theorem 4.4, also on page 58, it follows that it is unique.

[III] Such an edge e is called *signpost* in [6].

Let $\mathcal{L} = \{l_1, \dots l_{n-1}\}$ be again a set of vertical lines such that point $p_i \in P$ lies in the vertical slab between l_{i-1} and l_i, with $2 \leqslant i \leqslant n-1$. Point p_1, the leftmost, lies in the unbounded vertical slab to the left of l_1, and p_n, the rightmost, lies in the unbounded vertical slab to the right of l_{n-1}. For a pseudo-triangulation S of P let $\mathcal{P}(S) = \{pt(l_i, S) \mid l_i \in \mathcal{L}\}$. The following result is the equivalent of Theorem 4.6 on page 63 for T-paths and triangulations:

Theorem 4.8. *Let S be a pseudo-triangulation with vertex set P. Then $\mathcal{P}(S)$ is enough to characterize S.*

Proof. We will prove something stronger, namely, we will prove that *every* edge of a pseudo-triangulation S is an edge of some PT-path in $\mathcal{P}(S)$, this clearly implies the theorem. Observe that to prove the stronger statement we just have to prove that for any given edge e of S there exists a line l in $\mathcal{L}$ such that e is good w.r.t. l, or if there is no line of $\mathcal{L}$ that e is good with respect to, then we have to show that there is a line l of $\mathcal{L}$ such that e is used to connect two consecutive good edges of S w.r.t. l, that is, e is an edge of the common face of S^* that those two consecutive good edges of S w.r.t. l are part of. By a suitable rotation of the plane we will assume w.l.o.g. that *every* conceivable vertical line contains at most one point of P.

Let e be an edge of S. If e is an edge of $\mathcal{CH}(P)$ then there is clearly at least one line $l \in \mathcal{L}$ that intersects e, and thus it makes e the very first or the very last edge of $pt(l, S)$. Now assume that e lies strictly in the interior of $\mathcal{CH}(P)$ and let $\overline{\triangle}, \underline{\triangle}$ be the two pseudo-triangles that e is part of. By convention we will assume that a vertical line intersecting e intersects $\overline{\triangle}$ immediately *above* e, and intersects $\underline{\triangle}$ immediately *below* e.

In pseudo-triangulations, as in triangulations, the notion of flipping an edge exists: This time a flip exchanges the diagonal of a pseudo-quadrilateral by its other diagonal, however, for pseudo-quadrilaterals it is not always true that both its diagonals intersect, see Figures 4.26 and 4.28, while for triangulations that is always the case. Thus, both diagonals could appear in the same *non-pointed* pseudo-triangulation, nevertheless, in a pseudo-triangulation only one of them appears at a time, since the presence of both destroys either planarity or pointedness. We will thus inspect two cases, depending on whether the dual edge e' of e in the pseudo-quadrilateral $\square = \overline{\triangle} \cup \underline{\triangle}$ intersects e or not.

If e and e' intersect, let l be the vertical line containing their intersection point, see Figures 4.26 and 4.27. The reader can easily verify that the supporting lines of the edges $\overline{e}$ of $\overline{\triangle}$ and $\underline{e}$ of $\underline{\triangle}$, intersected by l right above and below e, intersect the supporting line of e on different sides of l, making e good w.r.t. l. It remains to argue what happens if $l \notin \mathcal{L}$, which can easily be the case. If $l \notin \mathcal{L}$ then l lies in the vertical slab between a pair of lines $l_{i-1}, l_i \in \mathcal{L}$, and p_i is the only point of P that also lies in that slab. Thus we can continuously sweep l in one direction as to make it coincide with either l_{i-1} or l_i without destroying any argument.

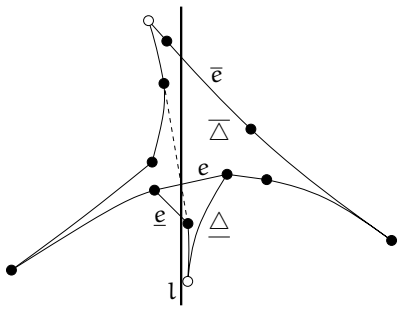

Figure 4.26 – The flip edge e' of e is shown dashed. If those two edges intersect, the e is good w.r.t. line l. The two vertices of $\square$ opposite to e are shown in white.

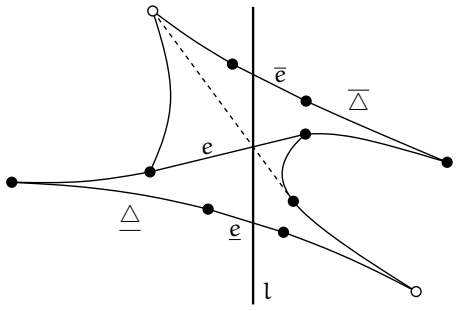

Figure 4.27 – Another possibility for $\square$.

If e and e' do not intersect, let us assume that there is no vertical line l contained in the vertical slab defined by e such that the supporting lines of the edges $\overline{e}, \underline{e}$ intersect the supporting line of e on different sides of l, otherwise e is good w.r.t. to l, see Figure 4.28. We will assume that the intersections between those supporting lines happen to the left of any vertical line that intersects e, see Figure 4.29.

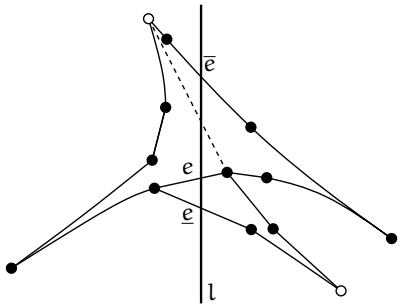

Figure 4.28 – If e and e' do not intersect, the pseudo-triangles of $\square$ can be oriented such that there is still a line l that e is good with respect to.

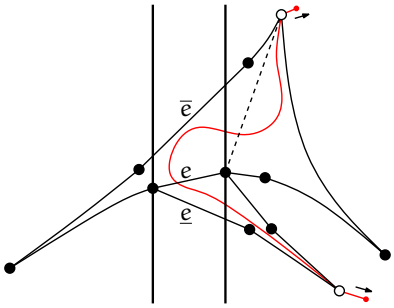

Figure 4.29 – If the red path is pulled from its ends in the direction shown by the arrows, until its length is minimal, we end up having a geodesic path between the opposite vertices, where e' is the only new edge.

Therefore we have to prove that e is actually used to connect two consecutive good edges of S w.r.t. some line that does not intersect e. Since e and e' do not intersect, it must be the case that e and e' share one vertex, this is because a flip can be seen

as a *geodesic path*[IV] between the two corners of $\square$ opposite to e. This geodesic path coincides with the boundary of $\square$ except at exactly one edge, which is the flip e' of e. Since this path does not properly intersect e, but connects two points on different sides of the supporting line of e, it must happen that one endpoint of e is part of the path, which is exactly the place where e' helps to complete the geodesic path, see Figure 4.29.

Let $p = p_i$ be the vertex of e that is also shared by e'. Note that p is the only point of P contained in the vertical slab defined by $l_{i-1}, l_i \in \mathcal{L}$. Also, observe that only one of those two lines intersects e, so let us assume w.l.o.g. that l_{i-1} is the one that intersects e. The configuration at which we arrive can be seen in Figure 4.30. Another configuration arises when the other vertex of e is the one shared by e'; the configuration would be mirror-reflected to the one presented here.

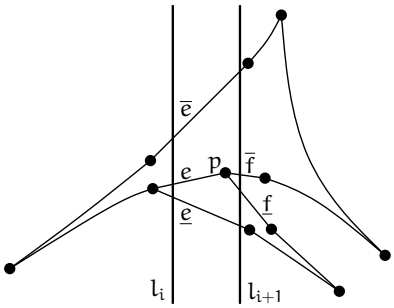

Figure 4.30 – Point p is the only point contained in the vertical slab between l_i, l_{i+1}. The configuration, if non-degenerate, must locally look like this.

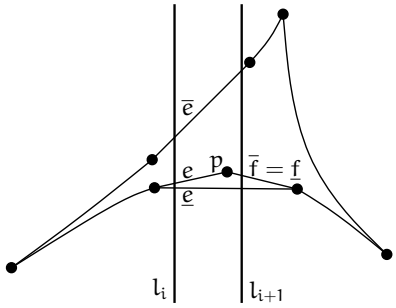

Figure 4.31 – If $\square$ is degenerate, then the configuration looks like this.

Let $\overline{f}, \underline{f}$ be the other edges of $\overline{\triangle}, \underline{\triangle}$ adjacent to p respectively. We claim that $\underline{f}$ is good w.r.t. l_{i+1}: If $\square$ is non-degenerate, then $\overline{f} \neq \underline{f}$, as displayed in Figure 4.30. In such a case observe that $\underline{e}, \underline{f}$ and $\overline{f}$ intersect l_{i+1} consecutively, the latter two intersect to the left of l_{i+1}, at p, and the supporting lines of the former two intersect to the right of l_{i+1}, which proofs the claim in this case. If $\square$ is degenerate, as displayed in Figure 4.31, then $\overline{f} = \underline{f}$, and thus $\overline{e}, \overline{f}$ and $\underline{e}$ intersect l_{i+1} consecutively. Here, the latter two share an endpoint to the right of l_{i+1}, and the supporting lines of the former two intersect to the left of l_{i+1}, this makes $\overline{f} = \underline{f}$ good again w.r.t. l_{i+1}. At this point observe that regardless of the case, the part of $\overline{\triangle}$ to the right of l_{i+1} cannot be used in $pt(l_{i+1}, S,)$ to connect $\underline{f}$ with the good edge w.r.t. l_{i+1} that lies above $\underline{f}$, since that part along with l_{i+1} does not form a pseudo-triangle, as the definition of a PT-path requires. Thus the part of $\overline{\triangle}$ to the left of l_{i+1} will be used in $pt(l_{i+1}, S)$, but that means that e will also be part of that PT-path, which concludes the proof. ■

[IV] A geodesic path between two points in a region R is the shortest path between the points that stays in R, including its boundary.

Hence, as for T-paths, every pseudo-triangulation S of P has a *unique* set $\mathcal{P}(S)$. Let $\Pi(l, P) = \{pt(l, S) \mid S \text{ is a pseudo-triangulations of } P\}$ be the set of *all* PT-paths w.r.t. to separating line l. What is now of interest to us is the opposite. Does every tuple $\{\pi_1, \ldots, \pi_{n-1}\}$ of pairwise non-crossing PT-paths define a *unique* pseudo-triangulation? Where $\pi_i \in \Pi(l_i, P)$ and $l_i \in \mathcal{L}$. The analogous statement for triangulations was clear, however, pseudo-triangulations might require more explanation. The answer is yes, as long as the union $\bigcup_{1 \leqslant i \leqslant n-1} \pi_i$ is pointed. To see this, just observe that if that union is pointed, then it can be completed to a pseudo-triangulation S_1 by adding edges, while keeping planarity and pointedness, see Theorem 4.1. Assume there is another pseudo triangulation S_2 that can be obtained from the union of the PT-paths π_i by adding edges in a different way. Observe that every PT-path π_i, $1 \leqslant i \leqslant n-1$, keeps being a PT-path of S_1, S_2 since the additional edges do not break planarity or pointedness. Thus by Theorem 4.8 there is no other option but $\bigcup_{1 \leqslant i \leqslant n-1} \pi_i = \mathcal{P}(S_1) = \mathcal{P}(S_2)$. But in the proof of that theorem we actually showed that *every* edge of S_1, S_2 is in some PT-path in $\mathcal{P}(S_1), \mathcal{P}(S_2)$ respectively, thus $S_1 = S_2$.

Thus the number of pointed pseudo-triangulations of P equals the number of different sets $\mathcal{P}(S)$ that we can find on P. The algorithm for counting pseudo-triangulations is the same as the algorithm for counting triangulations presented in the previous section, so we just have to define the sets the algorithm works on. Also, the proof of correctness will remain essentially the same, we will just point out what the differences are.

By previous discussions we know that a tuple $\{\pi_1, \ldots, \pi_{n-1}\}$ of PT-paths, where $\pi_i \in \Pi(l_i, P)$ and $l_i \in \mathcal{L}$, defines a pseudo-triangulations iff those PT-paths are pairwise non-crossing and their union is pointed. As before, we will use the term *compatible* for such a pointed and pairwise non-crossing set of PT-paths. We can now define the following set:

$$\mathcal{T}(\pi_j) = \{\{\pi_1, \ldots, \pi_{j-1}\} \mid \{\pi_1, \ldots, \pi_j\} \text{ is compatible, and } \pi_i \in \Pi(l_i, P), l_i \in \mathcal{L}\}$$

By the discussion above we have that the number of pointed pseudo-triangulations of P is exactly $|\mathcal{T}(\pi)|$, where π is the *unique* PT-path of P w.r.t. $l_{n-1} \in \mathcal{L}$.

Finally, and for completeness, for each $\pi' \in \Pi(l_{i+1}, P)$ and each $\pi \in \Pi(l_i, P)$, with $l_i, l_{i+1} \in \mathcal{L}$, we define:

$$\lambda(\pi') = \{\pi \in \Pi(l_i, P) \mid \pi \text{ is compatible with } \pi'\}$$
$$\mu(\pi) = \{\pi' \in \Pi(l_{i+1}, P) \mid \pi' \text{ is compatible with } \pi\}$$

The notation $\Pi(\cdot,\cdot), \mathcal{T}(\cdot), \lambda(\cdot)$ and $\mu(\cdot)$ is the same as the one used in § 4.2 for T-paths, but the definitions here reflect that we are now dealing with PT-paths instead.

Since the sweep line algorithm for counting pseudo-triangulations is the same as the one for counting triangulations, we just have to show how to obtain $\Pi(l_{i+1}, P)$, as well as $|\mathcal{T}(\pi')|$ for every $\pi' \in \Pi(l_{i+1}, P)$, having stored $\Pi(l_i, P)$ and $|\mathcal{T}(\pi)|$ for every $\pi \in \Pi(l_i, P)$, where $l_i, l_{i+1} \in \mathcal{L}$ are two consecutive *event points* of the sweep line algorithm. This, as for T-paths, will be accomplished by doing *local changes* to every PT-path $\pi \in \Pi(l_i, P)$, which we explain next. From this local changes we directly obtain $\Pi(l_{i+1}, P)$ as well as $\lambda(\pi')$ for each $\pi' \in \Pi(l_{i+1}, P)$. Thus, obtaining $|\mathcal{T}(\pi')|$ is easy since $|\mathcal{T}(\pi')| = \sum_{\pi \in \lambda(\pi')} |\mathcal{T}(\pi)|$. We will later prove that $\lambda(\pi')$ can be correctly computed in time $O\left(n^6 \cdot t_i\right)$, where $t_i = |\Pi(l_i, P)|$. Therefore the overall running time of the algorithm is $\sum_{l_j \in \mathcal{L}} O\left(n^6 \cdot t_j\right) \leqslant O\left(n^7 \cdot t\right)$, where $t = \max\{t_j\}$.

Let us now explain what the local changes in general look like. Let $p = p_{i+1} \in P$ be the point lying between lines $l_i, l_{i+1} \in \mathcal{L}$. As for T-paths, the only obstacle of *every* PT-path π of $\Pi(l_i, P)$ to be a PT-path π' of $\Pi(l_{i+1}, P)$ is p. The changes are mostly equivalent (in form) to the ones for T-paths but this time they are more complicated. We have two possibilities, depending on whether π has p as a vertex or not. Let us see each one in turn:

(1) If $\pi \in \Pi(l_i, P)$ has p as vertex we have more sub-cases depending on whether p lies inside $\mathcal{CH}(P)$ or on $\mathcal{CH}(P)$, and whether p is the convex vertex of an empty pseudo-triangle bounded by l_i or not. Let us see:

- If p lies strictly inside $\mathcal{CH}(P)$ let us first assume that p is also the convex vertex of an empty pseudo-triangle of π bounded by l_i. This case is equivalent to the one for triangulations displayed in Figure 4.16 on page 67. The situation is as displayed in Figures 4.32 and 4.33 with solid lines. Let e, f be the good edges of π w.r.t. l_i right below and above p respectively, and let e', f' be the good edges of π w.r.t. l_i adjacent to p such that e, e', f', f are ordered bottom-up along l_i. Let $\triangle$ ($\triangle'$) be the empty pseudo-triangle of π to the left of l_i having e, e' (f, f') as edges and bounded by l_i. If e and f share their right endpoint, then a PT-path $\pi' \in \Pi(l_{i+1}, P)$ can be produced using only adjacencies from the original PT-path $\pi \in \Pi(l_i, P)$, see Figure 4.33. This situation can easily be detected.

  If e and f do not share their right endpoint, then the situation is in general as displayed in Figure 4.32. The local changes we are looking for are produced by every point $\alpha \in P$ such that the dotted adjacencies shown in Figure 4.34 produce a PT-path $\pi' \in \Pi(l_{i+1}, P)$ with the property that $\pi \cup \pi'$ is pointed.

  So, let us explain more carefully how these changes are really produced. Let $\mathcal{I}$ be the interval of l_{i+1} seen by p having the edges of π as obstacles. The visibility cone of p towards l_{i+1} is shown dashed in Figures 4.32 and 4.33. Observe that every α used for a change has a visibility ray to $\mathcal{I}$. So having the

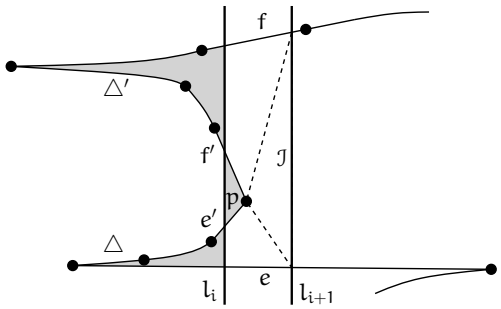

Figure 4.32 – Here e and f do not share the right endpoint.

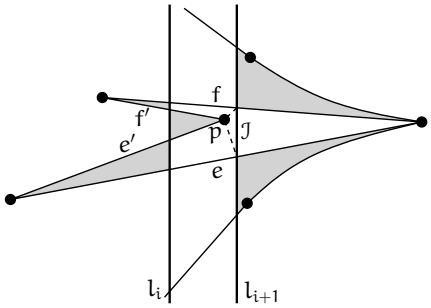

Figure 4.33 – In this case a PT-path $\pi' \in \Pi(l_{i+1}, P)$ can be produced using only adjacencies from the original PT-path $\pi \in \Pi(l_i, P)$.

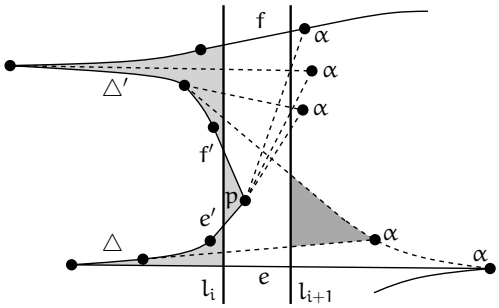

Figure 4.34 – All points α can be used to produce a PT-path $\pi' \in \Pi(l_{i+1}, P)$.

edges of π as obstacles, obtain a list A of all points to the right of l_{i+1} having a visibility ray to $\mathcal{J}$. This can be done in total time $O\left(n^2 \log(n)\right)$, see [62]. Let $\alpha \in A$. We will assume that we have actually computed a visibility cone to $\mathcal{J}$ with apex at α. We then regard α as the apex of an empty pseudo-triangle bounded by π' (to be constructed) and l_{i+1}, see the dark gray region to the right of l_{i+1} with apex at one of the α's in Figure 4.34. The same α can give rise to different PT-paths of $\Pi(l_{i+1}, P)$, see Figure 4.35.

So the way we discern between all the PT-paths of $\Pi(l_{i+1}, P)$ that can be obtained from a single $\alpha \in A$ is as follows: Shoot a visibility ray ρ from α to $\mathcal{J}$ that is fully contained in the empty pseudo-triangle delimited by l_{i+1} that α is apex of, the dashed lines of Figure 4.35. From the intersection point between ρ and $\mathcal{J}$ create two paths $\rho_\downarrow, \rho_\uparrow$ following $\mathcal{J}$ towards e and f respectively, so $\rho_\downarrow$ goes down, and $\rho_\uparrow$ goes up. Once e and f are reached, follow the adjacencies of π towards the leftmost convex vertex v, v' of $\triangle$, $\triangle'$ respectively. Paths $\rho_\downarrow, \rho_\uparrow$ are shown in red in Figure 4.35. Now, the adjacencies that are joining $\pi \in \Pi(l_i, P)$ with α are nothing but two shortest

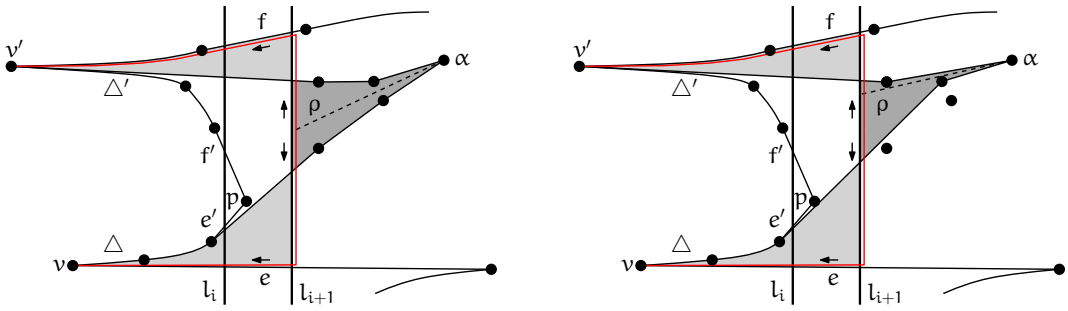

Figure 4.35 – Two different possibilities for adjacencies connecting α to $\pi \in \Pi(l_i, P)$. Each gives a different PT-path of $\Pi(l_{i+1}, P)$.

paths $\widetilde{\rho_\downarrow}, \widetilde{\rho_\uparrow}$ between α and v, v' respectively, the former homotopic to $\rho_\downarrow \cup \rho$ and the latter homotopic to $\rho_\uparrow \cup \rho$. Just imagine that if $\rho_\downarrow \cup \rho$ and $\rho_\uparrow \cup \rho$ are two strings between α and v, v' respectively, then pulling them as to make them of shortest length, having the points of P as obstacles, will give the adjacencies connecting α to π, and thus complete the adjacencies of $\pi' \in \Pi(l_{i+1}, P)$.

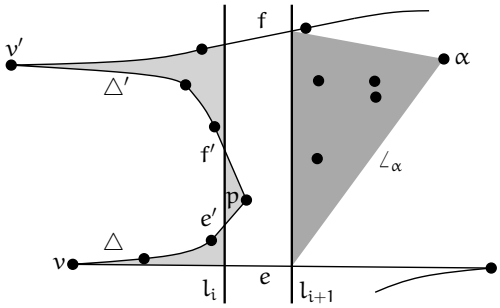

Figure 4.36 – The visibility cone $\angle_\alpha$ (to the right of l_{i+1}) is shown in dark gray.

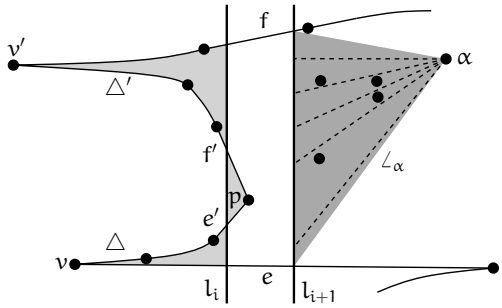

Figure 4.37 – Each of the dashed lines defines an homotopy class.

Thus, in order to construct all PT-paths of $\Pi(l_{i+1}, P)$ that can be obtained from $\alpha \in A$, we have to exhaust all its possibilities. This is done as follows: Consider the visibility cone $\angle_\alpha$ to $\mathcal{I}$ with apex at α, shown in dark gray in Figure 4.36. If $\angle_\alpha$ is empty, then any visibility ray ρ to $\mathcal{I}$ inside $\angle_\alpha$ will do to create $\rho_\downarrow$ and $\rho_\uparrow$. As a consequence of the emptiness of $\angle_\alpha$, point α will spawn only one PT-path of $\Pi(l_{i+1}, P)$. Otherwise, sort the points of P inside $\angle_\alpha$ angularly around α (clockwise). Now shoot visibility rays $\rho_0, \ldots, \rho_k$ from α to $\mathcal{I}$ such that between any two consecutive visibility rays there is exactly one point of P, and use each visibility ray $\rho = \rho_i$, $0 \leqslant i \leqslant k$, to create paths $\rho_\downarrow$ and $\rho_\uparrow$ as described before. Since $\angle_\alpha$ is non-empty, ray ρ defines the homotopy

class that paths $\widetilde{\rho_\downarrow}, \widetilde{\rho_\uparrow}$ belong to. Thus, potentially, every ray ρ_i, $0 \leqslant i \leqslant k$, could give a PT-path of $\Pi(l_{i+1}, P)$. Figure 4.38 shows a configuration where a visibility ray does not produce a PT-path $\pi' \in \Pi(l_{i+1}, P)$ where α is a convex vertex of an empty pseudo-triangle of π' bounded by l_{i+1}.

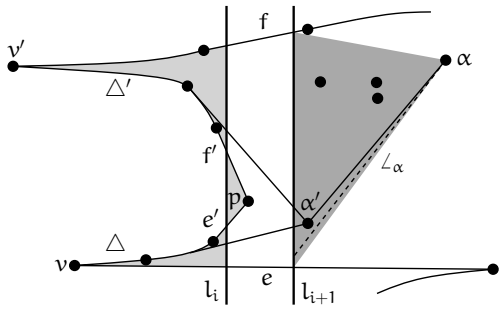

Figure 4.38 — Visibility ray shown in dashed defines the homotopy that the adjacencies connecting α with π should follow. In this case the created path is not a PT-path of $\Pi(l_{i+1}, P)$ where α is a convex vertex. It would be nevertheless a PT-path of $\Pi(l_{i+1}, P)$ where α' is a convex vertex. This path will be detected when processing α'.

So, given $\alpha \in A$, obtaining the points of P lying inside $\angle_\alpha$, and their sorted order around α, can be done in $O(n \log(n))$ time. For each visibility ray $\rho \in \{\rho_i\}_{i=0}^k$, we can construct the paths $\rho_\downarrow, \rho_\uparrow$ in $O(n)$ time, and the shortest homotopic paths $\widetilde{\rho_\downarrow}, \widetilde{\rho_\uparrow}$ can be computed in $O\left(n^2\right)$, see [17] and references therein. Thus, we spend $O\left(n^3\right)$ time to exhaust all possibilities for α, and it can spawn $O(n)$ different PT-paths of $\Pi(l_{i+1}, P)$. Doing this for every element of A takes $O\left(n^4\right)$ time in total, where also the total number of PT-paths produced is $O\left(n^2\right)$. Clearly, by construction, the union of each PT-path $\pi' \in \Pi(l_{i+1}, P)$ constructed this way from a PT-path $\pi \in \Pi(l_i, P)$ is non-crossing and pointed.

If p is *not* the convex vertex of an empty pseudo-triangle of π bounded by l_i, then the situation is essentially like displayed in Figure 4.39. A similar construction can be done that looks like mirror-reflected. Using the same notation as before, the empty pseudo-triangles $\triangle, \triangle'$ lie on different sides of l_i and l_{i+1}. Also, only the edges e, f are good w.r.t. l_i and l_{i+1}, edge e' is good w.r.t. l_i only, and edge f' is good w.r.t. l_{i+1} only. In the "mirror-reflected" construction, edge f' is the one that is good w.r.t. l_i, and edge e' is the one that is good w.r.t. l_{i+1}.

Observe that we cannot extend $\triangle$ to an empty pseudo-triangle bounded by l_{i+1} since point p would be a convex vertex of such extension, and thus

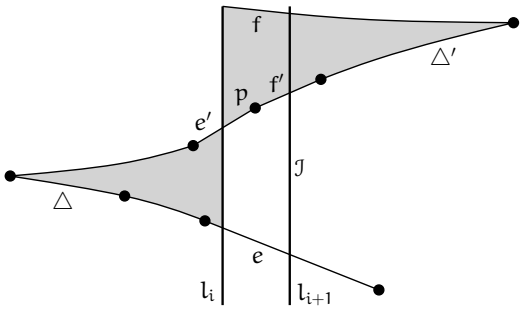

Figure 4.39 – The symmetric configuration in which $\triangle$ and $\triangle'$ lie on opposite sides is also possible.

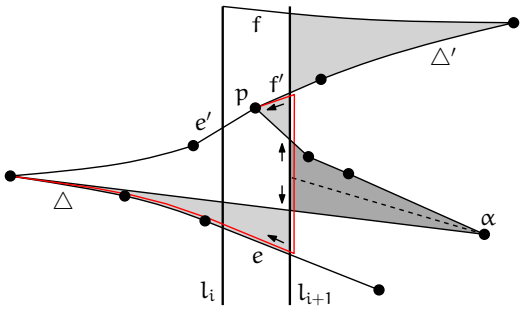

Figure 4.40 – The red lines connect α to p and to the leftmost convex vertex of $\triangle$ via the visibility ray shown dashed. These two paths define the homotopy the local changes must follow.

that extension would be a pseudo-quadrilateral, see Figure 4.39. No such a problem occurs with $\triangle'$.

The way we deal with this situation is very similar to the previous case. Let $\mathcal{J}$ and A be as before. For every $\alpha \in A$ define again the visibility cone $\angle_\alpha$, and construct the set of rays $\{\rho_i\}_{i=0}^k$ as well. For $\rho \in \{\rho_i\}_{i=0}^k$, define the path $\rho_\downarrow$ just as before. This time, however, define $\rho_\uparrow$ as the path that connects the intersection point of ρ and l_{i+1} with p by following l_{i+1} up to edge f', and then f' to p. We now compute the two shortest paths $\widetilde{\rho_\downarrow}, \widetilde{\rho_\uparrow}$ homotopic to $\rho_\downarrow \cup \rho, \rho_\uparrow \cup \rho$ respectively. So again we exhaust all possibilities of every point in A. The time remains $O\left(n^4\right)$ in total, and again the number of PT-paths of $\Pi(l_{i+1}, P)$ produced is $O\left(n^2\right)$. If α is the right endpoint of e' or of f', then one of the shortest homotopic paths overlaps with the adjacencies of π, and thus it must be ignored in the resulting PT-path of $\Pi(l_{i+1}, P)$. The reader can use Figure 4.40 by imaging pulling α to the right endpoint of e'. Another example of such a degeneracy will be seen later on.

Observe again that pointedness and planarity is kept.

- If p lies on $\mathcal{CH}(P)$ then one possible configuration is as the one shown in Figure 4.41, in which p is the last, or first, vertex of $\pi \in \Pi(l_i, P)$. Another possibility arises when p is the second, or second-to-the-last, vertex of π. Which shortest homotopic paths should be computed should be clear from the figure by now.

(2) If π *does not* have p as a vertex, then p must necessarily lie inside $\mathcal{CH}(P)$. The situation is in general as displayed in Figure 4.42. In this case there are two kinds of

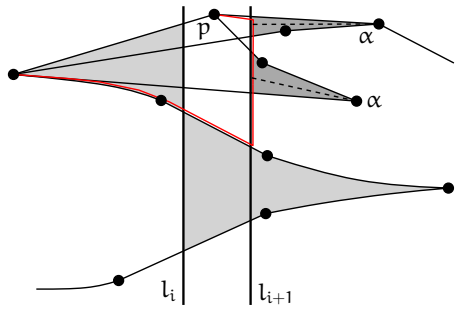

Figure 4.41 – In this case p lies on $\mathcal{CH}(P)$ and its degree in $\pi \in \Pi(l_i, P)$ is exactly one. Two possibilities using two different α's are shown.

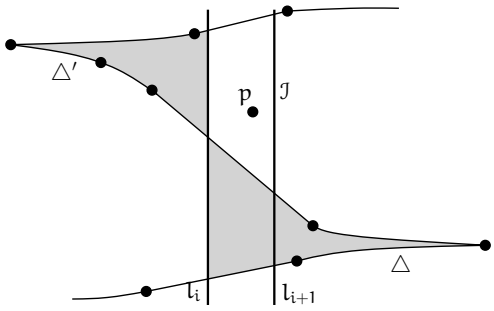

Figure 4.42 – Although p is not a vertex of $\pi \in \Pi(l_i, P)$, it must be part of some $\pi' \in \Pi(l_{i+1}, P)$ since the empty pseudo-triangle $\triangle'$ of π cannot be extended further.

local changes that can be made; one kind is produced by a single point $\alpha \in P$, and the other kind is produced by pairs of points $\alpha, \beta \in P$, see Figures 4.43 and 4.44 for a reference.

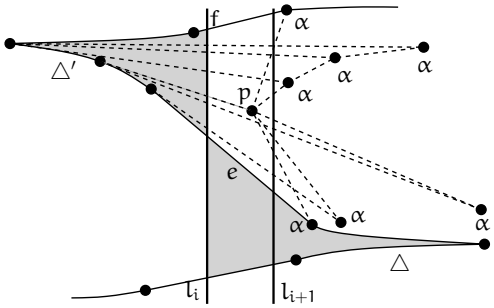

Figure 4.43 – Changes are produced only by one point α.

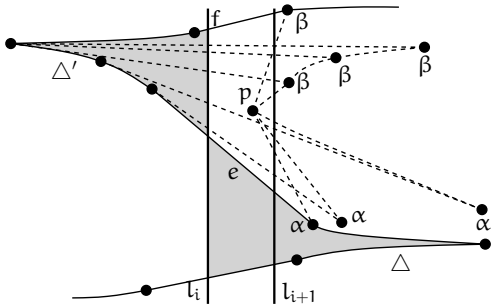

Figure 4.44 – Changes are now produced by pairs of points α, β.

Let J, A be defined as before. Let us see each kind of local changes in turn. For the local changes produced by just one point $\alpha \in A \subset P$, the PT-paths of $\Pi(l_{i+1}, P)$ produced look like the ones in Figure 4.45.

Using the same ideas as before, of following red paths, the adjacencies of α in a PT-path of $\Pi(l_{i+1}, P)$ are two shortest paths homotopic to the two red paths shown in Figure 4.45, one going up and the other going down, and the visibility ray from α to J, shown dashed in Figure 4.45. Using the visibility cone $\angle_\alpha$ we can again exhaust all possibilities for α in $O(n^3)$ time, and thus we exhaust all of A in $O(n^4)$ time, producing $O(n^2)$ PT-paths of $\Pi(l_{i+1}, P)$ in total. As a remark, observe that if α is the right endpoint of edge f or e, then one of the shortest homotopic paths overlaps completely with adjacencies of $\pi \in \Pi(l_i, P)$, this path

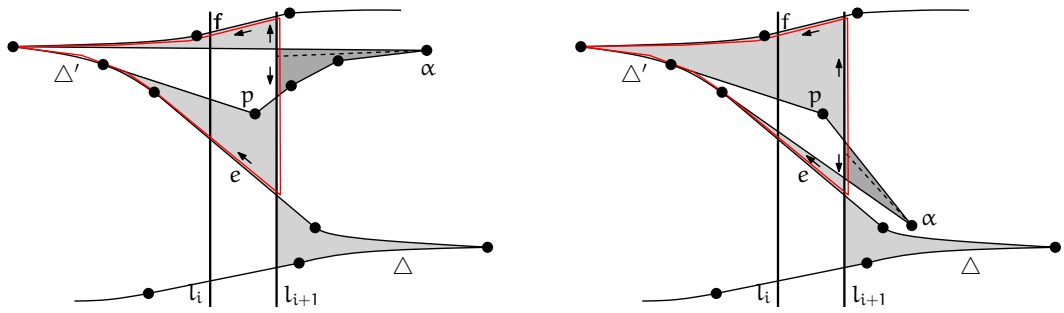

Figure 4.45 – Two different PT-paths of $\Pi(l_{i+1}, P)$ produced by two different points.

can be ignored, and then the produced PT-path of $\Pi(l_{i+1}, P)$ would look like the one in Figure 4.46, where the path of π connecting α with the leftmost convex vertex of $\triangle'$ is the one ignored.

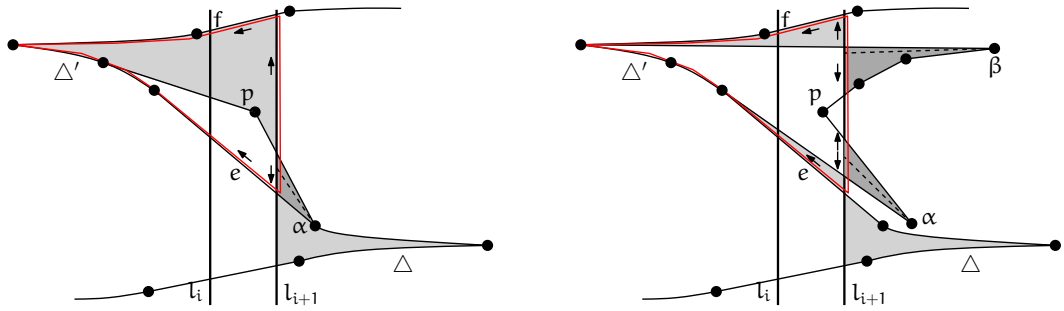

Figure 4.46 – A particular case occurs if α coincides with an endpoint of e or of f.

Figure 4.47 – Combining the PT-paths shown in Figure 4.45 we obtained yet another PT-path of $\Pi(l_{i+1}, P)$, we just had to remove the adjacencies of p that make it non-pointed.

As for the local changes produced by pairs of points $\alpha, \beta \in A \subset P$, the PT-paths of $\Pi(l_{i+1}, P)$ produced look like the one shown in Figure 4.47. If we have constructed the PT-paths produced by a single $\alpha \in A$, then we can construct the paths produced by pairs $\alpha, \beta \in A$ by combining the local changes applied to α, with all the local changes applied to β. For example, the PT-path $\pi' \in \Pi(l_{i+1}, P)$ shown in Figure 4.47 is obtained from the PT-paths of Figure 4.45, by removing the adjacencies at p that do not make it pointed. So, when combining changes we have, of course, to be careful about pointedness and planarity of the construction, which takes not much more effort to verify.

Since the total number of different PT-paths produced by α, β is $O(n)$, by combining them we will obtained no more than $O(n^2)$ PT-paths. Thus, by going through every pair $\alpha, \beta \in A$, the total number of PT-paths of $\Pi(l_{i+1}, P)$ produced is $O(n^4)$, and all this can be achieved in $O(n^6)$ time, since combining a pair can be achieved in $O(n^2)$ time.

This concludes the explanation of the local changes that need to be made to PT-paths as we sweep.

As for T-paths, the local changes of PT-paths can be seen in reverse order, as going from line l_{i+1} to l_i, so we will use again the notation $\pi \leftrightarrow \pi'$ to denote the fact that $\pi' \in \Pi(l_{i+1}, P)$ is produced from $\pi \in \Pi(l_i, P)$ in one direction, so $\pi' \in \mu(\pi)$, and π is produced from π' in the reverse direction, so $\pi \in \lambda(\pi')$.

We can now prove the following result which is the equivalent to Lemma 4.6 on page 68 for T-paths:

Lemma 4.7. *Given $\Pi(l_i, P)$, every PT-path of $\Pi(l_{i+1}, P)$ is produced by the local changes just explained. Moreover, for each $\pi \in \Pi(l_i, P)$, the cardinality of $\mu(\pi)$ is $O(n^4)$, and we can correctly compute $\lambda(\pi')$, for each $\pi' \in \Pi(l_{i+1}, P)$, in time $O(n^6 \cdot t_i)$, where $t_i = |\Pi(l_i, P)|$.*

Proof. For the first part of the lemma an argument as the one we used for the first part of Lemma 4.6 can be used. We can check that given any PT-path $\pi' \in \Pi(l_{i+1}, P)$ we can always obtain a PT-path $\pi \in \Pi(l_i, P)$ by locally changing π', and thus every PT-path of $\Pi(l_{i+1}, P)$ is produced by the relation $\pi \leftrightarrow \pi'$. The second part, the correct computation of $\lambda(\pi')$ for every $\pi' \in \Pi(l_{i+1}, P)$, also follows by a similar argument as the one we did in Lemma 4.6 in the corresponding part, that is, $\pi \not\leftrightarrow \pi'$ implies that π and π' properly cross.

Finally, the size of $\mu(\pi)$ and the time it takes to compute $\lambda(\pi')$, for *every* $\pi' \in \Pi(l_{i+1}, P)$, follows from the explanations done while explaining the local changes of PT-paths. Hence the lemma follows. ■

This concludes the proof of Theorem 4.5.

4.4 Discussion and conclusions

The problem of "algorithmically" counting crossing-free structures defined on given sets of points is directly related to the problem of generating random crossing-free structures. For example, we might be interested in producing a triangulation of a given set of points

P uniformly at random, that is, *every* triangulation of P must appear with probability $\frac{1}{|\mathcal{F}_T(P)|}$. This allows us to study structural properties of an "average" triangulation of P, for example, to check how many of its vertices have a given degree, or to verify what fraction of its vertices has a degree of certain parity. This could allow us to make conjectures on triangulations and to try to prove them using induction, for which the base cases can be checked by computer.

Methods to produce random triangulations are known, for example, in [2] a method is explained that produces random triangulations using the divide-and-conquer algorithm therein presented. For the sweep line algorithms that we just presented another method can be used (due to a different paradigm): Assume we want to generate a random triangulation, generating random pointed pseudo-triangulations is the same. Remember that we sweep from left to right, so we store for every event point l_i, $1 \leqslant i \leqslant n-1$, and for every T-path π found w.r.t. l_i, the cardinality of $\mathcal{T}(\pi)$, which is the number of structures to the left of l_i that are compatible with π. We construct a random triangulation by sweeping in reverse order once the algorithm has finished the counting. Since there is only one path w.r.t. l_{n-1} we choose it. Going from l_{i+1} to l_i, $1 \leqslant i < n-1$, and having fixed a path π_{i+1} w.r.t. l_{i+1}, we choose a path π_i w.r.t. l_i with probability $\frac{|\mathcal{T}(\pi_i)|}{|\mathcal{T}(\pi_{i+1})|}$. By the time we arrive at l_1 we have generated a triangulation with probability:

$$1 \cdot \frac{|\mathcal{T}(\pi_{n-2})|}{|\mathcal{T}(\pi_{n-1})|} \cdot \frac{|\mathcal{T}(\pi_{n-3})|}{|\mathcal{T}(\pi_{n-2})|} \cdots \frac{|\mathcal{T}(\pi_1)|}{|\mathcal{T}(\pi_2)|} = \frac{|\mathcal{T}(\pi_1)|}{|\mathcal{T}(\pi_{n-1})|} = \frac{1}{|\mathcal{F}_T(P)|}$$

since there is only one T-path w.r.t. l_1. The downside of this method is that we need to compute the number of triangulations of P beforehand.

There is nevertheless a different method that seems to be quite good in practice, this method works by randomly flipping edges of a triangulation (with a pseudo-triangulation it would be the same). It is known that this method leads to a random triangulation in polynomial time for sets of points in convex position, see [58, 54]. Note, however, that since the number of triangulations of a convex polygon is a Catalan number, a triangulation generated uniformly at random can be obtained in optimal linear time, see [35] and references therein. For general sets of points nothing is known about the convergence of the random flipping procedure. This is a very interesting and challenging open problem.

4.4.1 Conclusions

In this chapter we have presented algorithms to compute the number of triangulations and pseudo-triangulations of a given set of points P. Both algorithms are rather simple

and they are based on T-paths, PT-paths and the sweep line paradigm. We also provided the first non-trivial upper bound for the number of T-paths of P w.r.t. to a given separating line. Unfortunately, this number turned out to be rather large, $O(9^n)$. We believe that the real upper bound for this number is closer to 4^n, which remains being very large nevertheless. However, we are not aware of any configuration of points, large enough, having as many T-paths as triangulations. This has previously been supported by experiments and proven for many known configurations of points. We will also see our own experiments in Chapter 6.

It seems that our T-path algorithm really is counting triangulations in time sub-linear in the number of triangulations, so we believe that this algorithm is still very interesting from the theoretical point of view. We suspect the same about our PT-path algorithm for counting pseudo-triangulation. An easy argument can be done to show that these algorithms are, in any case, no worse than enumeration algorithms. Although this sounds pessimistic, there are algorithms for which such an argument cannot be done.

The holy grail of counting triangulations is to prove polynomial time or #P-hardness. So far we have failed to prove any of them. Thus, the most interesting open questions at this moment are (in ascending order of importance): (1) For n large enough, is it true that there are always asymptotically more triangulations (pseudo-triangulations) than T-paths (PT-paths) w.r.t. a given separating line? (2) Is it possible to count triangulations (pseudo-triangulations) in sub-exponential time? Or even count approximately in polynomial time? (3) Is the problem of counting triangulations (pseudo-triangulations) in P, or is it #P-complete? Each one of these questions looks very challenging.

CHAPTER 5

COUNTING TRIANGULATIONS AND OTHER CROSSING-FREE STRUCTURES VIA ONION LAYERS

In previous chapters we have met three particular kinds of crossing-free structures on P, namely; quadrangulations, triangulations, and pseudo-triangulations. Other well-known crossing-free structures that can be defined on P are matchings, and spanning cycles. We remind the reader that a matching of P is a crossing-free structure on P where *every* vertex has degree at most one, and a spanning cycle of P is a single simple polygon with n sides whose vertex set is P.

As we did for triangulations and pseudo-triangulations in the previous chapter, we can also define $\mathcal{F}_C(P)$ and $\mathcal{F}_M(P)$ as the class of *all* spanning cycles, and *all* matchings of P respectively, and we can try to count their elements as well. Nevertheless, we have to keep in mind that those classes of crossing-free structures are also in general exponentially large. Their cardinality can be again of the form c^n for a constant c that depends on the class, for example, for $\mathcal{F}_C(P)$ is currently known that $c \leqslant 54.55$, see [76], and a configuration where $c \geqslant 4.65$ is known, see [61]. The interested reader can visit [78, 28] for an up-to-date list of bounds on all these, and also other classes, of crossing-free structures. The references therein give a good account of all listed bounds.

In this chapter we continue the design of counting algorithms for crossing-free structures.

5.1 Our contribution

We mentioned before that, although Theorem 4.3, see page 57, could potentially count triangulations in $o\left(|\mathcal{F}_T(P)|\right)$, and thus beat brute force enumeration, its running time might still be pretty large. This is because its running time depends on the number of T-paths that the algorithm encounters during its execution, and there are configurations having at least $\Omega(4^n)$ T-paths.

In this chapter we present, among other results, yet another new algorithm to count triangulations. Along with this new algorithm we also present algorithms to count the elements of $\mathcal{F}_C(P)$ and $\mathcal{F}_M(P)$, the classes of spanning cycles and matchings of P respectively. It is important to keep in mind that, so far, no algorithm is known that *always* beats enumeration on those classes.

In order to state the results we present in this chapter we need the following definition:

Definition 5.1 (Onion layers). Let P be a set of n points on the plane and let $\mathcal{CH}(P)$ denote its convex hull. We define the *onion layers* of P as follows: The first onion layer $P^{(1)}$ of P is $\mathcal{CH}(P)$. For $i > 1$, the i-th onion layer $P^{(i)}$ of P is defined inductively as $\mathcal{CH}\left(P \setminus \bigcup_{j=1}^{i-1} P^{(j)}\right)$. By "number of onion layers of P" we mean the number of *non-empty* onion layers of P.

Observe that the number of onion layers of *any* non-degenerate set of n points is *always* at most $\lceil \frac{n}{3} \rceil$. We are now able to state our results.

5.1.1 The new result on counting triangulations

Theorem 5.1 (V. Alvarez, R. Curticapean, K. Bringmann, S. Ray). *Let P be as before and let k be its number of onion layers. Then the exact value of $|\mathcal{F}_T(P)|$ can be computed in time $O\left(k^2 \cdot g\left(\frac{n}{k}\right)^n\right)$, where $g(x) = \left(\frac{x^3+3x^2+2x+2}{2}\right)^{\frac{1}{x}}$. Since $k \leqslant \lceil \frac{n}{3} \rceil$, this bound never exceeds $O^*(3.1414^n)$. This running time can alternatively be bounded by $n^{O(k)}$, which is polynomial for constant k.*

Thus the algorithm of the previous theorem has better worst-case behavior than that of Theorem 4.3, which is $O\left(9^n\right)$. Moreover, it has other nice properties:

- It is the *first* algorithm to be known that can compute the exact value of $|\mathcal{F}_T(P)|$ in *polynomial time* in at least some non-trivial cases.

- As stated before, for *every* set of n points, the size of $\mathcal{F}_T(P)$ can be lower-bounded by $\Omega(2.4^n)$, but it is widely believed that this bound can be improved to $\Omega\left(\sqrt{12}^n\right) \approx \Omega(3.464^n)$. If this stronger bound is true, then the algorithm of Theorem 5.1 would always count triangulations in time $O^*(3.1414^n) = o(|\mathcal{F}_T(P)|)$. Thus setting in the positive the answer of whether or not one can *always* count triangulations of set of points faster than enumerating them.

5.1.2 The results on counting other crossing-free structures

Moving away from triangulations, the other result that will be proven is the following:

Theorem 5.2 (V. Alvarez, R. Curticapean, K. Bringmann, S. Ray). *Let P be as before and let k be its number of onion layers. Then the exact values of $|\mathcal{F}_M(P)|$ and $|\mathcal{F}_C(P)|$ can be computed in $n^{O(k)}$ time.*

Thus again, as long as $k = O(1)$, the algorithms of Theorem 5.2 compute the said numbers in polynomial time, which then gives a partial answer to Problem 16 of The Open Problems Project, which asks whether $|\mathcal{F}_C(P)|$ can *always* be computed in polynomial time, see [29]. This time, however, we are not able to prove a running time of the sort c^n for large k, like in Theorem 5.1.

The general layout of the algorithms of Theorems 5.1 and 5.2 is similar to the one found in [12], where these ideas have been used for optimization problems.

We will divide the rest of the chapter is organized as follows: In § 5.2 we give a rough idea on how the algorithms work. In § 5.3 we prove Theorem 5.1, and in § 5.4 we prove Theorems 5.2. We close the chapter in § 5.5 with some conclusions.

5.2 A general framework for counting crossing-free structures

The overall idea of *all* our algorithms can be roughly described as follows. Suppose we want to count the elements of some particular class $\mathcal{F}$ of crossing-free structures on P. A set S of non-crossing edges on P is called a *separator* if the union of the edges in S splits the interior of $\mathcal{CH}(P)$, possibly along with $\mathcal{CH}(P)$, into at least two regions. In such a case we will say that S splits $\mathcal{CH}(P)$ into those regions. Now assume that there exists a set $\mathcal{S}$ of separators with the following properties: (1) Every element of $\mathcal{F}$ contains a *unique* separator $S \in \mathcal{S}$, and (2) we can "quickly" enumerate the members of $\mathcal{S}$. With a set of separators $\mathcal{S}$, the elements of $\mathcal{F}$ can be counted as follows: For each $S \in \mathcal{S}$, let $R_1^S, R_2^S, \ldots, R_t^S$ be the regions S splits $\mathcal{CH}(P)$ into. Recursively compute the number

n_i^S of elements of $\mathcal{F}$ of each region R_i^S. The number of elements of $\mathcal{F}$ containing S is then $N^S = \prod_{i=1}^{t} n_i^S$. Thus the total number of elements of $\mathcal{F}$ is simply $\sum_{S \in \mathcal{S}} N^S$. Of course, in the recursion, a set of separators is required in each R_i^S, and the efficiency of the algorithm depends heavily on the choice of $\mathcal{S}$. For example, one well-known family of separators $\mathcal{S}$ for triangulations is the set of T-paths, which we saw in Chapter 4. We will introduce some other families of separators, some of them with additional properties, however, for the time being we believe that this vague description of how the algorithms work conveys the main idea appropriately.

5.3 Counting triangulations using the onion layers

In this section we will present yet another new algorithm for counting triangulations that uses the onion layers of the given set of points P.

Now, for any $p \in P$, let $\ell(p)$ denote the index of the onion layer to which p belongs. Let us label the points $p \in P$ with distinct labels in $\{1, \ldots, n\}$ such that if $\ell(p) < \ell(q)$ then p also receives a label smaller than q. This is clearly possible. Figure 5.1 shows the onion layers of a set of 17 points and the labels assigned to them. From now on we will refer to the points of P by their labels *i.e.*, we will think of P as the set $\{1, \ldots, n\}$ and when we say "$p \in P$", we will mean the point with label p.

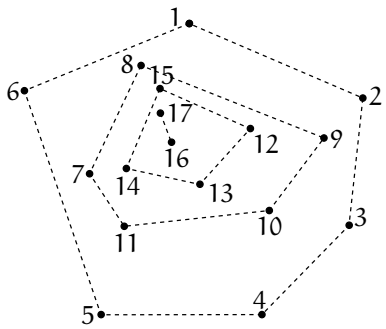

Figure 5.1 – Four onion layers.

Let T be any triangulation of P. For $p \in P \setminus P^{(1)}$, let $sn_T(p)$ be the smallest neighbor of p in T. Observe that any such point p has at least one neighbor q such that $\ell(q) < \ell(p)$ and therefore $sn_T(p) < p$. If $p \in P^{(1)}$, we set $sn_T(p) = p$. When T is clear from context, we will just write $sn(p)$ instead of $sn_T(p)$. We denote by sn-path$_T(p)$ the unique path $p = a_0, a_1, \ldots, a_m$ in T such that for each $0 \leqslant i < m$, we have that $a_{i+1} = sn(a_i)$ and $sn(a_m) = a_m$. We will also direct this path from a_0 towards a_m and call this the direction of "descent" since $\ell(\cdot)$ decreases along the path. Note that any sn-path consists

of at most one point from each onion layer and precisely one point from the first onion layer.

Let (p, q) be some edge in T and suppose that sn-path(p) ends at $p' \in P^{(1)}$ and sn-path(q) ends in $q' \in P^{(1)}$. There are two paths in T from p' to q' along $\mathcal{CH}(P)$, one in the clockwise direction and the other in the counter-clockwise direction. Each of these paths along with the edge (p, q) and the two sn-paths starting at p and q respectively, defines a region within $\mathcal{CH}(P)$. We call these two regions the sn-regions of (p, q). See Figure 5.2. Given any sn-region R, we refer to the number of triangles in any triangulation of R as the *size* of R. This is well defined since the number of triangles is the same regardless of the triangulation chosen.

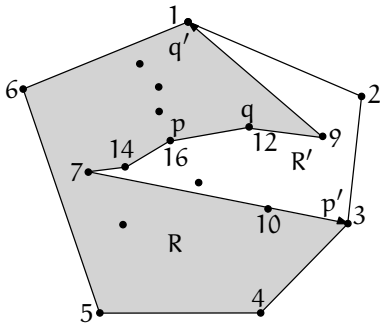

Figure 5.2 – R and R' are the sn-regions of (p, q).

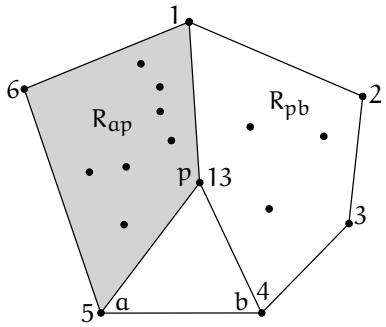

Figure 5.3 – R_{ap} and R_{pb} are the sn-regions of (a, p) and (p, b), respectively, that do not contain triangle apb.

Let ab be an edge on $\mathcal{CH}(P)$. Observe that in any triangulation, $\mathcal{CH}(P)$ is one of the sn-regions of (a, b), the other region being empty. In any triangulation T of P, there is precisely one triangle apb that the edge ab belongs to. Let R_{ap} be the sn-region of (a, p) that does not contain apb and similarly let R_{pb} be the sn-region of (p, b) that does not contain apb, see Figure 5.3.

5.3.1 The algorithm

Let ab be again an edge on $\mathcal{CH}(P)$. The core idea of our algorithm is as follows: We can easily enumerate all the points p such that the triangle apb appears in some triangulation. This is just the set Q of points p such that the triangle apb is free of other points of P. For every element p of Q, suppose that we can enumerate the sn-paths ρ of p over all triangulations of P. For ever pair (p, ρ), let $\mathcal{T}_{(p,\rho)} = \mathcal{T}_{(p,\rho)}(P)$ be the set of triangulations of P that contain the triangle apb and in which ρ is the sn-path of p.

If, for each such pair that we can obtain, we can compute $|\mathcal{T}_{(p,\rho)}|$, then we are done, since each triangulation of P must contain precisely one pair (p, ρ), adding the numbers over all pairs gives us the total number of triangulations.

Let us fix a pair (p, ρ) for which we would like to compute $|\mathcal{T}_{(p,\rho)}|$. The pair already defines the regions R_{ap} and R_{pb} for all triangulations in $\mathcal{T}_{(p,\rho)}$. Observe that any triangulation in $\mathcal{T}_{(p,\rho)}$ contains a triangulation T_{ap} of R_{ap} and a triangulation T_{pb} of R_{pb}, each of which satisfy the following sn-constraint: For each edge (q, r) in ρ there is no edge (q, s) in the triangulation (either T_{ap} or T_{pb}) such that $s < r$. Furthermore, putting together any pair of triangulations T_{ap} and T_{pb}, each satisfying the constraint, and the triangle apb gives a triangulation in $\mathcal{T}_{(p,\rho)}$. This observation follows from the fact that ρ is an sn-path of p in any triangulation of $\mathcal{T}_{p,\rho}$, and allows us to separately compute the number of (sn-constraint-satisfying) triangulations N_{ap} of R_{ap} and N_{pb} of R_{pb} whose product gives $|\mathcal{T}_{(p,\rho)}|$.

The numbers N_{ap} and N_{pb} are computed recursively. We will maintain the invariant that at any point in the recursion we are dealing with an sn-region of some edge. This is certainly true in the beginning since we start with an sn-region of the edge ab and also in the next step since we recurse on sn-regions defined by the edges (a, p) and (p, b) respectively. At any point, let us say that we are dealing with an sn-region R defined by an edge (x, y) and let ρ_x and ρ_y be the sn-paths starting at x and y respectively.

Now, we recurse almost exactly as we did before: We enumerate the set of points z such that the triangle xzy lies within R and is free of other points of P contained in R, see Figure 5.4. Furthermore, we ensure that if z happens to be a point in either ρ_x or ρ_y, and (z, t) is an edge in that sn-path, then both x and y are bigger than t. This way, we do not violate the sn-constraint. For each such z we enumerate the portions of sn-paths starting at z that lie within R. See Figure 5.4. Each such path splits the region R into regions R_{xz} and R_{zy} which are sn-regions defined by (x, z) and (z, y) respectively. Each of the regions R_{xz} and R_{zy} have sizes smaller than R, *i.e.*, fewer triangles in any triangulation. The recursion bottoms out when the size is $\leqslant 1$, in which case we know that there is exactly one triangulation. Note that even though we enumerate only the portions of the sn-paths of z that lie within R, these portions implicitly define the entire sn-path of z. This is because such a portion either ends at a point on the first onion layer in which case it is the entire sn-path, or at a point w on either ρ_x or ρ_y. The direction of descent along that sn-path, starting at w, is then the remaining portion of the sn-path of z.

One detail is still missing. How do we enumerate the portions of the sn-paths of z that belong to at least one triangulation of R? and the answer is: We will not do it. Instead, we enumerate a superset of paths which are *descending* in the sense that they start at z and each successive point is in a strictly upper layer (a layer containing points with smaller indices). Again, we only enumerate the portion of such paths that lie inside R

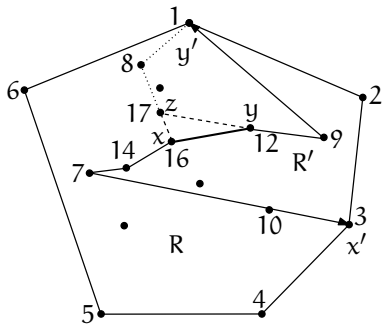

Figure 5.4 – R and R′ are the sn-regions of (x, y).

since the rest is implicitly defined. For any descending path that does not really belong to any triangulation of R, at least one of the regions R_{xz} or R_{zy} has no triangulations satisfying the sn-constraint. This will be detected somewhere down the recursion where we will not be able find any z satisfying the sn-constraint. At that point, we return 0 as the number of triangulations. Thus the algorithm still works in these cases.

There is one other ingredient that we add for efficiency: Memoization. Whenever we compute the number of triangulations of a certain sn-region that satisfy the sn-constraint dictated by the sn-paths defining the region, we store it in a hash table (or any other data structure). Now consider a *call graph* in which each node represents an sn-region and there is a directed edge from a region R to a region R′ if from R we make a recursive call to R′. The number of egdes in this graph is an upper bound[I] on the running time of the algorithm since, because of memoization, no edge is *traversed* more than once.

We will now prove an upper bound on the number of edges in the call graph. Each call from a region R to a region R′ can be charged to a triple of descending paths - two defining R and a third that, along with a triangle, splits R into two regions, one of which is R′. The triples (ρ_1, ρ_2, ρ_3) that are produced in the algorithm have the property that once two paths merge in the direction of descent, they never split again. This is ensured by the fact that we only enumerate the portions of the third descending path within the region R and the rest is implicitly defined, as noted before. Let ρ_2' be the portion of ρ_2 that does not have any point in common with ρ_1, and let ρ_3' be the portion of ρ_3 that does not have any point in common with either ρ_1 or ρ_2. The descending paths ρ_1, ρ_2' and ρ_3' are vertex disjoint, and along with some additional information they completely describe ρ_1, ρ_2 and ρ_3. The additional information that is required is whether, and where, ρ_2 merges with ρ_1, and whether, and where, ρ_3 merges with one of the other paths. If P has k onion layers, then each descending path has length at most k and therefore there are at most k ways that ρ_2' may merge with ρ_1, and at most $2k$ ways ρ_3'

[I] Up to a polynomial overhead arising from the construction and handling of sub-problems.

may merge with one of ρ_1 or ρ_2. Therefore, if U is an upper bound on the number of triples of vertex disjoint descending paths, then $2k^2U$ is an upper bound on the number of triples (ρ_1, ρ_2, ρ_3) as described above, and hence also an upper bound on the running time of the algorithm.

5.3.2 Number of vertex-disjoint triples of descending paths

Each descending path uses at most one vertex from every onion layer. Let $n_i = |P^{(i)}|$ be the size of the i-th onion layer. Let us count how many ways there are for any triple of paths to use at most one vertex each from this layer. There is one way for the triple of paths to skip this onion layer. There are n_i ways of choosing one point among the n_i which may then be used by any of the paths. This gives $3n_i$ ways for the three paths. There are $\binom{n_i}{2}$ ways to choose two points, and any two of the paths may use them. This gives $6\binom{n_i}{2}$ ways among the three paths. Finally there are $\binom{n_i}{3}$ ways of choosing three points, and there are three (not six) ways for the three paths to use one of these vertices. This is because these paths are non-crossing planar curves, and therefore the clockwise order of these paths along any $\mathcal{CH}\left(P^{(i)}\right)$ that intersects all three of them is the same for each i. The overall number of ways in which at most three points can be used from the i-th layer is therefore $f(n_i)$, where $f(x) = 1 + 3x + 6\frac{x(x-1)}{2} + 3\frac{x(x-1)(x-2)}{6}$.

The number of triples of vertex disjoint descending paths is therefore at most $U = \prod_{i=1}^{k} f(n_i)$. Since each n_i is a positive integer, and the function $f(\cdot)$ is log-concave for $x \geqslant 1$, the above product is maximized when each n_i is equal to $\frac{n}{k}$. This gives an upper bound of $f\left(\frac{n}{k}\right)^k = g\left(\frac{n}{k}\right)^n$, where $g(x) = f(x)^{\frac{1}{x}}$. Now, $g(x)$ is maximized for some value of x between 0 and 1 and is a decreasing function for $x \geqslant 1$. Since each onion layer except the k-th one must have at least three points, we have $U = O\left(g(3)^n\right)$. The fact that the k-th onion layer may have fewer than three points makes only a difference of a constant factor. Therefore the running time of the algorithm presented in this section is $O\left(k^2 g(3)^n\right) = O^*(3.1414^n)$. This concludes the proof of Theorem 5.1.

We want to point out that often the number of onion layers can be much smaller than the maximum possible $\lceil \frac{n}{3} \rceil$. For example, Dalal [26] has shown that if n points are chosen uniformly at random from a disk, then the expected number of onion layers of the resulting point set is $\Theta\left(n^{2/3}\right)$.

From a theoretical point of view, the algorithm presented in this section, sn-path algorithm for short, has a running time polynomial in n whenever the number of onion layers of P is constant. This is the first known algorithm for counting triangulations having this property. Also, its worst-case behavior is better than the one based on T-paths that we presented before, $O^*(3.1414^n)$ of the former against $O(9^n)$ of the latter.

The sn-path algorithm is an excellent candidate for having good experimental behavior as well, due to its polynomial-time instances. Towards the end of Chapter 6 we shall see how the sn-path performs experimentally against the fastest-known (in practice) algorithm for counting triangulations presented in [70], and also against our own T-path-based algorithm.

5.4 Counting other crossing-free structures

In this section we show how the core ideas of the previous two algorithms for counting triangulations, the T-path-based of section 4.2 and sn-path-based of section 5.3, can be "modified" or "adapted" so that we can count other crossing-free structures on P. We start by showing how the ideas of the sn-path algorithm can be used to develop a general framework that helps to count crossing-free structures in general. We use this framework to count perfect matchings and spanning cycles defined by P.

5.4.1 Counting matchings and spanning cycles

Assume we want to count crossing-free matchings spanned by P. Clearly any matching can be completed to a triangulation by adding edges, and thus we might want to try the technique used for counting triangulations: Take a set $\mathcal{S}$ of separators and for each $S \in \mathcal{S}$ count the matchings in triangulations containing S, and finally add this up over all $S \in \mathcal{S}$. In any matching M that can be completed to a triangulation containing S, each vertex in S is either unmatched, or it is matched to a vertex within some R_i^S, or it is matched to another vertex in S. We can *annotate* each separator S with this information. When counting, for each $S \in \mathcal{S}$, we iterate over all annotations of S, and take care to be consistent with the current annotation when recursing into the sub-problems.

This simple algorithm fails because some matchings M could be contained in a triangulation that could contain several, say $s_M > 1$, separators and would thus fool our algorithm to count M exactly s_M times. If $s_M = s$ were a constant over all matchings we would not have this problem, however, we are not aware of any set of separators $\mathcal{S}$ with this property.

There is however a way in which we can modify the simple algorithm so that we can count each matching exactly once: We embed each matching M into a unique triangulation $T \supset M$. Given a family $\mathcal{S}$ of separators for the triangulations of P, we associate a unique $S \in \mathcal{S}$ to each matching. For concreteness, let us associate to each M the constrained Delaunay triangulation (CDT) $\triangle^M$ *constrained* to contain M, which we briefly describe next.

Constrained Delaunay Triangulation: The constrained Delaunay triangulation (CDT) $\triangle^S$ of P was first introduced in [24]. Formally, it is the triangulation T of P containing S such that no edge e in $T \setminus S$ is flippable in the following sense: Let $\triangle_1, \triangle_2$ be triangles of P sharing e. The edge e is flippable if and only if $\square = \triangle_1 \cup \triangle_2$ is convex, and replacing e with the other diagonal of $\square$ increases the smallest angle of the triangulation of $\square$. One of the most important properties of constrained Delaunay triangulations is its uniqueness if no four points of P are cocircular. Thus, under standard non-degeneracy assumptions, there is a unique CDT for any given set of mandatory edges. For a good study on constrained Delaunay triangulations we suggest the book [46] by Ø. Hjelle and M. Dæhlen.

For our counting purposes we will assume that no four points of P are cocircular. This can easily be taken care of by perturbing P. We can now go back to our simple algorithm for counting matchings and revise it as follows: Whenever we recurse, in each sub-problem we only count matchings M such that $S \subseteq \triangle^M$, where $S \in \mathcal{S}$ is a separator. If this last condition can be satisfied locally in each sub-problem, *i.e.*, choices in one sub-problem do not depend on choices in others, we are done. While not every $\mathcal{S}$ admits such a locality condition, some do as we will see next.

5.4.2 Triangular paths

We assume again that P has k onion layers. For every point $p \in P$ (on layer $P^{(i)}$ which is not the first layer) we fix in advance a ray ρ_p which emanates from p and does not intersect the interior of $\mathcal{CH}\left(P^{(i)}\right)$.

For any triangulation T of P there is a unique triangle $\triangle_p = p, q_1, q_2$ adjacent to p and intersecting ρ_p. Let q_p be the smaller of q_1 and q_2, using the same labeling as before. Clearly q_p lies in a layer lower than the one containing p. Let $p_0, p_1, \ldots, p_r$ be the sequence so that $p_0 = p$, $p_{i+1} = q_{p_i}$, $\forall\, 0 \leqslant i < k$, and p_r lies on the first layer. We call $P_p(T) := \bigcup_i \triangle_{p_i}$ the *triangular path* of p w.r.t. T, and we call p_r the *last point* of $P_p(T)$. See Figure 5.5.

The triangular path $P_p(T)$ is uniquely defined for any triangulation. Moreover, for distinct triangulations T_1 and T_2, $P_p(T_1), P_p(T_2)$ are either identical or they intersect properly: Let i be the first position where $\triangle_{p_i}(T_1) \neq \triangle_{p_i}(T_2)$, then those two triangles intersect, as they both are adjacent to p, intersect ρ_p and have interiors free of points in P. We are now ready to finish the algorithm for counting matchings.

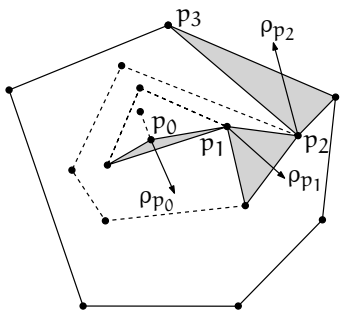

Figure 5.5 – Triangular path P_p starting in onion layer $P^{(4)}$. Onion layers are shown in dashed. P_p can be extended to a triangulation T, in such a case P_p will be unique for T.

Algorithm for counting matchings

Given a matching M, let $\triangle^M$ be the CDT of M. By our assumption of no four cocircular points, this CDT is unique for M. We annotate $\triangle^M$ as follows:

- each vertex v of $\triangle^M$ is annotated with a number m_v that represents the vertex of M that v is matched to. If m_v is 0 say, then we know that v is not matched in M.
- each edge e of $\triangle^M$ is annotated with a bit b_e that indicates whether e belongs to M or not.

Let us denote by $\overline{\triangle}^M$ the annotated version of $\triangle^M$. Let S be a separator contained in $\triangle^M$ that splits $\mathcal{CH}(P)$ into regions $R_1, \ldots, R_t$. Separator S inherits all the information from $\overline{\triangle}^M$. We additionally keep track of whether m_v is any of the adjacencies of v in S, for each vertex $v \in S$. If not then we set m_v to the index $1 \leqslant i \leqslant t$ of the region R_i the matching vertex of v falls into (it must necessarily be a region having v as a vertex of its boundary). The separator thus annotated will be denoted by $\overline{\triangle}^M_S$.

We say that an annotated constrained Delaunay triangulation is *legal* if and only if it is identical to $\overline{\triangle}^M$, for some matching M. Since there is a one-to-one correspondence between matchings and legal constrained Delaunay triangulations, our goal is to count the latter.

Our algorithm is essentially the same as for counting triangulations: Instead of sn-paths we use annotated triangular paths. We start with an edge ab on $\mathcal{CH}(P)$, and enumerate the set of points p such that the triangle apb is free of other points of P. For each such p, the triangle apb along with the triangular path starting at p forms a separator, see Figure 5.6. We enumerate such separators and all possible annotations for each one of them. Each such annotated separator splits $\mathcal{CH}(P)$ into two smaller

regions in which we recurse. In each recursive sub-problem we count (legal) annotated constrained Delaunay triangulations consistent with the annotated separator.

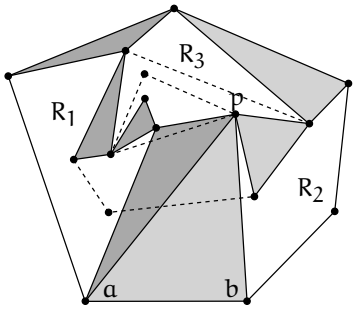

Figure 5.6 – In the first call of the algorithm, the triangular path shown in dark gray is created. It divides the problem into regions $R_1 \cup R_3$ and R_2. A call for the latter creates the triangular path shown in light gray. Annotations are not shown for simplicity.

The reason for which we use triangular paths instead of simple sn-paths is the following: No edge in a separator, formed by a triangular path, lies on the boundary of more than one sub-problem. This allows us to verify flippability of edges separately in each sub-problem. If an edge belonged to more than one sub-problem, then the flippability of this edge would depend on the choices made in each sub-problem, thus introducing dependency between these sub-problems.

As in the case for counting triangulations, we use *memoization*. The running time as before is dominated by the number of triples of annotated triangular paths. The size of each triangular path is $O(k)$, thus there are clearly at most $n^{O(k)}$ triangular paths. Also, trying all possible annotations per triangular path leads to no more than $n^{O(k)}$ annotations per triangular path, as can be easily checked. Hence there are $n^{O(k)}$ annotated triangular paths, and also $n^{O(k)}$ triples of annotated triangular paths. The overall running time is thus $n^{O(k)}$, which even considers the polynomial overheads arising from checking flippability of edges and inclusion of points into sub-problems. This concludes one part of Theorem 5.2.

The annotations required for counting matchings are not very complicated, but for many other counting problems this is a highly non-trivial task. An example of more involved annotations is given next, where we consider the problem of counting spanning cycles.

Algorithm for counting spanning cycles

Counting spanning cycles is more complicated than counting matchings. What we will actually do is that, instead of counting spanning cycles, we will count *rooted and oriented* spanning cycles. Given any cycle, we make it rooted by designating a *starting vertex*, and we make it oriented by assigning an *orientation*- clockwise or counterclockwise. We then number the vertices in the cycle from 1 to n, beginning at the starting vertex (which is the root of the cycle), and continuing along the assigned direction. We also direct the edges along this direction. This way, each spanning cycle is counted exactly $2n$ times. At the end we divide the computed number by $2n$ to get the desired number. In the remainder we use the term HamCycle for rooted and oriented spanning cycles.

Given a HamCycle H let $\triangle^H$ be the CDT of H. We annotate $\triangle^H$ as follows:

- each vertex v of $\triangle^H$ is annotated with $(\text{pos}_v, \text{prev}_v, \text{next}_v)$, where pos_v is the number assigned to v in H, prev_v is the vertex lying immediately before v in H, and next_v is the vertex lying immediately after v in H.
- each edge e in $\triangle^H$ is annotated with a bit b_e that indicates whether e belongs to H or not.

As in the case for matchings, the annotated $\triangle^H$ will be denoted by $\overline{\triangle}^H$. Let S be again a separator contained in $\triangle^H$ that splits $\mathcal{CH}(P)$ into regions $R_1, \ldots R_t$. Separator S inherits the following information from $\overline{\triangle}^H$: Each vertex $v \in S$ inherits pos_v from $\overline{\triangle}^H$. If prev_v and next_v are already adjacent to v in S then this information is also inherited. If prev_v is absent in S then v is annotated with the index i, $1 \leqslant i \leqslant t$, of the region R_i that prev_v falls in. The same holds for next_v. Each edge e of S carries the annotation it has in $\overline{\triangle}^H$. The separator S of Δ^H thus annotated will be denoted by $\overline{\triangle}^H_S$.

The algorithm, as the reader might be thinking right now, is no other than the algorithm for counting matchings. The only difference are the annotations, they encode a different problem. Thus again, the number of triangular paths is $n^{O(k)}$. The number of annotations per triangular path stays $n^{O(k)}$, and hence the total running time will stay at $n^{O(k)}$, including again the other polynomial overheads. This finishes the proof of Theorem 5.2.

5.5 Conclusions

In this chapter we have presented algorithms to count triangulations, crossing-free matchings and crossing-free spanning cycles of a given set of points P. All algorithms use the onion layers of P and the divide-and-conquer paradigm.

The algorithm to count triangulations presented in this chapter has the best provable worst-case running time as of this writing, $O^*(3.1414^n)$. We consider important to note that configurations of points having from $\Omega(3.464^n)$, see [72], to $\Omega(8.65^n)$, see [33], triangulations are known. Thus the algorithm presented in this chapter counts triangulations faster than enumeration algorithms in at least those cases. Also, this algorithm has polynomial-time instances whenever the number of onion layers of the given set of points is constant. We will see in the next chapter that, experimentally, for up to 3 onion layers this algorithm outperforms the algorithm of [70], which is reported to be extremely fast in practice. At this point the most interesting open question is the following: Is it true that *every* set of n points, n being large enough, spans at least $\Omega(3.464^n)$ triangulations? If this is true, then the algorithm we presented in this chapter *always* counts triangulations in $o(|\mathcal{F}_T(P)|)$.

Speaking about counting other kinds of crossing-free structures, we showed again two algorithms. The first one could be seen more as a framework for counting essentially "every" kind of crossing-free structures, since it depends on a labeling scheme, which is the hardest part of the algorithm to come up with. This "framework" implies algorithms with running times of the sort $n^{O(k)}$, where k is the number of onion layers of the given set of points, which again, for fixed k implies polynomial time. Algorithms like these were not known before for this kind of problems. This gives a partial answer to Problem 16 of The Open Problems Project, which asks whether $|\mathcal{F}_C(P)|$ can *always* be computed in polynomial time, see [29]. These counting algorithms also allow us to generate crossing-free matchings and spanning cycles uniformly at random. The latter being a problem that has attracted the attention of researchers for almost 20 years, in the form of generating random simple polygons on P, which is nothing but a crossing-free spanning cycle of P. Since our algorithms are based on the divide-and-conquer paradigm, we can adapt the method explained in [2] to produce such random structures, once the counting has been done. Other methods to generate random simple polygons without having to count are known, for example, in [13] many heuristics for polygons are presented. There the authors reported that (uniform) random generation can be done in polynomial time when the random polygon is star-shaped, but in the general case the algorithms therein presented are either unpractical or unable to generate uniformly at random.

Finally, it is worth noting that although we could have tried to come up with an annotation scheme for pseudo-triangulations, and thus we could have obtained another algorithm to count pseudo-triangulations using the onion layers, the resulting algorithm would have had a running time of the sort $n^{O(k)}$. This running time is in general not better than $O^*(t)$, at least from the theoretical point of view, since for the latter we have $t = O(c^n)$, for some constant c, while for the former k can get linearly large. The most important open problem here is whether the number of matchings and spanning cycles can *always* be computed in polynomial time.

CHAPTER 6

MISCELLANEOUS RESULTS ON COUNTING TRIANGULATIONS

In this chapter we conclude the study on counting algorithms that we started in Chapter 4. The topic turned out to be very vast, and as it often occurs in research, one stumbles upon results on the same topic but of rather different nature. This chapter contains two such results, one is an algorithm for approximate counting triangulations, and the other is a hardness result of a very particular instance of the problem of (exactly) counting triangulations. Along with these two results we also show experiments comparing our two algorithms for counting triangulations, of Chapters 4 and 5, with the algorithm presented in [70], which is supposed to be *very* fast in practice.

6.1 Our contribution

The first result, whose proof will be shown in § 6.2, is the following:

Theorem 6.1 (V. Alvarez, K. Bringmann, S. Ray, R. Seidel). *Let* P *be a set of* n *points on the plane, and let* $c \in \mathbb{R}$ *be such that* $|\mathcal{F}_T(P)| = c^n$. *Then a number* Λ *can be computed in time* $2^{o(n)}$ *such that* $c^n \leqslant \Lambda \leqslant c^{n+o(n)}$.

This first result is an approximation algorithm for counting the triangulations of P. While the approximation factor of Λ is rather big, there is something very important to note here: The approximation is within the same order of growth of the real value

of $|\mathcal{F}_T(P)|$, that is, the base of the exponentially large computed value is still c, more precisely $c \leqslant \Lambda^{\frac{1}{n}} \leqslant c^{1+o(1)} \leqslant (1+o(1))c$. Also, this approximation can be computed in sub-exponential time, which, at least theoretically, is asymptotically faster than the worst-case instances of the algorithms of Chapters 4 and 5. This is certainly very appealing.

The second result is a hardness result. Observe that the running times of the algorithms of Theorems 5.1 and 5.2 can be stated as $n^{f(k)}$, for some function f that does not depend on n. With regard to parameterized complexity it is natural to ask if these running times can be improved to something of the sort $g(k) \cdot n^{O(1)}$, for some function g independent of n, thus proving that our problems belong to the FPT complexity class. Which is the class of fixed-parameter tractable problems. However, the techniques involved in the algorithms of Theorems 5.1 and 5.2 are general enough to solve harder problems, such as the following:

Restricted-Triangulation-Counting-Problem: Given a set of points P and a subset of edges E over P, count the triangulations of P that use only edges from E.

The hardness result, whose proof will be presented in § 6.3, is the following:

Theorem 6.2 (V. Alvarez, R. Curticapean, K. Bringmann, S. Ray). *The* Restricted-Triangulation-Counting-Problem *is W[2]-hard if the parameter is considered to be the number of onion layers of* P. *This result even holds for the problem of just deciding the* existence *of a restricted triangulation.*

The book by J. Flum and M. Grohe, see [37], is a standard reference for Parameterized Complexity Theory, where the classes FPT and W[2] are defined. For now, however, it suffices to say that the separation FPT $\neq$ W[2] is widely believed. Thus an algorithm with a running time of the sort $g(k) \cdot n^{O(1)}$ is most likely not attainable for the Restricted-Triangulation-Counting-Problem. This might be an indication that we may have to exploit the particular structure of the problems in order to obtain fixed-parameter tractable algorithms for counting crossing-free structures, in the general non-restricted case.

In § 6.4 the experimental results mentioned before will be shown. We close the chapter in § 6.5 with some conclusions.

6.2 Counting triangulations approximately

In this section, as in the previous chapter, we use separators as the main ingredient of our algorithm. Here, however, they come in the form of simple cycles, that is, cycles

that do not cross themselves. The following result[I] shows that separators with this form exist, and that they have very appealing properties:

Theorem 6.3 (G. L. Miller, H. N. Djidjev, S. M. Venkatesan). *Let* T *be a triangulation of a set of* n *points on the plane such that the unbounded face is a triangle. Then there exists a simple cycle* C *of size at most* $\sqrt{4n}$*, which separates the set* A *of vertices of* T *in its interior from the set* B *of vertices of* T *in its exterior, such that the number of elements of each one of* A *and* B *is always at most* $\frac{2n}{3}$.

Observe however that the result of Miller does not imply that *every* triangulation of a set of points contains a *unique* simple cycle separator. One can easily come up with examples in which one triangulation contains more than one simple cycle separator. The important part here is that *every* triangulation contains *at least* one simple cycle separator. This is the reason behind the use of simple cycle separators for approximate counting.

The idea for a counting algorithm now suggests itself: We will enumerate all possible simple cycles separators C of size at most $\sqrt{4n}$ that we can find in the given set P. We will then recur in each of the parts A and B of Theorem 6.3 delimited by C[II], and add or multiply the numbers we thus obtain accordingly.

With the previous algorithm we clearly over-count *all* triangulations of P. However, there are technicalities that we have to overcome. For starters, Miller's result holds only if the unbounded face of T is also a triangle, so if we add a dummy vertex v_∞ outside $\mathcal{CH}(P)$, along the adjacencies between v_∞ and the vertices of $\mathcal{CH}(P)$, to make the unbounded face a triangle, we can apply Miller's result. Once a simple cycle with the dividing properties of a separator is found, by the deletion of v_∞ we are left with a separator that is either the original cycle that we found, if v_∞ does not appear as a vertex of the separator, or a path otherwise. So when guessing a separator it suffices to consider that it might be a path instead of a cycle. This brings us to the second technical issue. As we go deeper in the recursion we might create "holes" in P whose boundaries are the separators that we have considered thus far. So the recursive problems are polygonal regions, possibly with holes, containing points of P. Therefore, when guessing a separator, cycle or path, we have to keep in mind that the separator might be "extended" by the boundaries of the holes it encounters[III]. This extensions are not guessed nevertheless, they are just detected and added, so this does not modify the size of the sets we guess for a separator in a sub-problem.

What is now interesting is the running time of the algorithm as well as the quality of its approximation. We will devote the rest of this section to these two subjects.

[I] Originally presented in [57] by G. L. Miller, and improved in [31] by H. N. Djidjev and S. M. Venkatesan.
[II] Thus separator C also forms part of the two sub-problems.
[III] Think of it as the separator dividing the hole it encounters into many parts.

6.2.1 Quality of approximation

We first prove the following lemma:

Lemma 6.1. *Let $\mathcal{F}_T(P)$ be the set of triangulations of a set P of n points. Then* all *separators, simple cycles or paths, among* all *the elements of $\mathcal{F}_T(P)$ can be computed in time $2^{o(n)}$.*

Proof. We know by Theorem 6.3 and the discussion above that *every* element of $\mathcal{F}_T(P)$ contains at least one separator C, simple cycle or path. Moreover, the size of C is at most $\sqrt{4n}$, thus searching by brute-force will do the job. We can enumerate all the subsets of P of size at most $\sqrt{4n}$ along with their permutations. A permutation is what tells us how to connect the points of the guessed subset. We can then verify if the constructed simple cycle, or path, fulfills the dividing properties of a separator, as specified in Theorem 6.3.

It is not hard to check that the total number of guessed subsets and their permutations is $2^{O(\sqrt{n}\log(n))}$. Identifying whether a cycle, or a path, is indeed a separator can be done in polynomial time. So the total time spent remains being $2^{O(\sqrt{n}\log(n))}$. ■

By the proof of the previous theorem we also obtain that the number of simple cycle separators cannot be larger than $2^{O(\sqrt{n}\log(n))}$. Since at *every* stage of the recursion of the counting algorithm *no* triangulation of P can contain more than the total number of simple cycle separators found at that stage, we can express the over-counting factor of the algorithm by the following recurrence:

$$S(P,\Delta) = \sum_{C} S(A \cup C,\Delta) \cdot S(B \cup C,\Delta) \leqslant 2^{O(\sqrt{n}\log(n))} \cdot S(A \cup C^*,\Delta) \cdot S(B \cup C^*,\Delta)$$

Where the summation is over all separators C available at the level of recursion, $A \cup C$, $B \cup C$ are the sub-problems as explained before, C^* is the cycle that maximizes $S(A \cup C,\Delta) \cdot S(B \cup C,\Delta)$ over all C, and Δ is a stopping condition which means that whenever the sub-problem we recur in contains $\leqslant \Delta$ points, we stop the recursion, and we compute the number of triangulations exactly instead, so we have a boundary condition $S(Q,\Delta) = 1$ whenever $|Q| \leqslant \Delta$.

The product of $S(A \cup C,\Delta)$ and $S(B \cup C,\Delta)$ means that we are combining *all* triangulations of $A \cup C$ with *all* triangulations of $B \cup C$. We can now write:

$$S'(P,\Delta) := \log(S(P,\Delta)) \leqslant O\left(\sqrt{n}\log(n)\right) + S'(A \cup C^*,\Delta) + S'(B \cup C^*,\Delta)$$

Our goal now is to prove the following lemma:

Lemma 6.2. $S'(P,\Delta) = O\left(\left(\frac{n}{\sqrt{\Delta/3}} - \sqrt{n}\right)\log(\Delta)\right)$, *for a suitably chosen value of Δ.*

Proof. We will proceed by induction over $P' \subseteq P$ of size $m \leqslant n$, so we have:

$$\begin{aligned} S'(P', \Delta) &\leqslant O\left(\sqrt{m}\log(m)\right) + S'(A \cup C^*, \Delta) + S'(B \cup C^*, \Delta) \\ &\leqslant O\left(\sqrt{m}\log(m)\right) + c\left(\frac{m_1}{\sqrt{\frac{\Delta}{3}}} - \sqrt{m_1} + \frac{m_2}{\sqrt{\frac{\Delta}{3}}} - \sqrt{m_2}\right)\log(\Delta) \end{aligned} \tag{6.1}$$

Where m_1, m_2 are the sizes of the sub-problems $A \cup C^*$ and $B \cup C^*$ respectively, and c is some large enough positive constant. Thus, observe that we can express $m_1 \leqslant \alpha m + \sqrt{4m}$ and $m_2 \leqslant \beta m + \sqrt{4m}$, such that: (1) α, β are constants that depend on the instance, so $\alpha = \alpha(A \cup C^*)$ and $\beta = \beta(B \cup C^*)$, (2) $0 < \beta \leqslant \alpha \leqslant \frac{2}{3}$[IV], and (3) $\alpha + \beta = 1$.

Now let us for the moment focus on the term $\frac{m_1}{\sqrt{\Delta/3}} - \sqrt{m_1} + \frac{m_2}{\sqrt{\Delta/3}} - \sqrt{m_2}$ of 6.1:

$$\begin{aligned} \frac{m_1}{\sqrt{\frac{\Delta}{3}}} - \sqrt{m_1} + \frac{m_2}{\sqrt{\frac{\Delta}{3}}} - \sqrt{m_2} &= \frac{m_1 + m_2}{\sqrt{\frac{\Delta}{3}}} - \sqrt{m_1} - \sqrt{m_2} \\ &\leqslant \frac{\alpha m + \sqrt{4m} + \beta m + \sqrt{4m}}{\sqrt{\frac{\Delta}{3}}} - \sqrt{m_1} - \sqrt{m_2} \\ &\leqslant \frac{m + 4\sqrt{m}}{\sqrt{\frac{\Delta}{3}}} - \sqrt{m_1} - \sqrt{m_2} \leqslant \frac{m + 4\sqrt{m}}{\sqrt{\frac{\Delta}{3}}} - \sqrt{\alpha m} - \sqrt{\beta m} \\ &\leqslant \frac{m + 4\sqrt{m}}{\sqrt{\frac{\Delta}{3}}} - \sqrt{m}\left(\sqrt{\alpha} + \sqrt{\beta}\right) \leqslant \frac{m + 4\sqrt{m}}{\sqrt{\frac{\Delta}{3}}} - \sqrt{m}(1 + \varepsilon) \end{aligned}$$

The last inequality is obtained by minimizing $\sqrt{\alpha} + \sqrt{\beta}$. Since we mentioned before that $0 \leqslant \beta \leqslant \alpha \leqslant \frac{2}{3}$, then the minimum of $\sqrt{\alpha} + \sqrt{\beta}$ is attained $(\alpha, \beta) = \left(\frac{2}{3}, \frac{1}{3}\right)$, and is strictly larger than one, so $\varepsilon > 0$. Now, if Δ is large enough, then $\frac{4\sqrt{m}}{\sqrt{\Delta/3}} \ll \varepsilon\sqrt{m}$, so $\frac{4\sqrt{m}}{\sqrt{\Delta/3}} - \varepsilon\sqrt{m} = -d_1\sqrt{m}$, for some positive constant d_1. Thus we can continue as follows:

$$\frac{m_1}{\sqrt{\frac{\Delta}{3}}} - \sqrt{m_1} + \frac{m_2}{\sqrt{\frac{\Delta}{3}}} - \sqrt{m_2} \leqslant \frac{m + 4\sqrt{m}}{\sqrt{\frac{\Delta}{3}}} - \sqrt{m}(1 + \varepsilon) \leqslant \frac{m}{\sqrt{\frac{\Delta}{3}}} - \sqrt{m} - d_1\sqrt{m} \tag{6.2}$$

Combining equations 6.1 and 6.2 we obtain:

$$\begin{aligned} S'(P', \Delta) &\leqslant O\left(\sqrt{m}\log(m)\right) + c\left(\frac{m}{\sqrt{\frac{\Delta}{3}}} - \sqrt{m} - d_1\sqrt{m}\right)\log(\Delta) \\ &\leqslant c\left(\frac{m}{\sqrt{\frac{\Delta}{3}}} - \sqrt{m}\right)\log(\Delta) + O\left(\sqrt{m}\log(m)\right) - c \cdot d_1\sqrt{m}\log(\Delta) \end{aligned}$$

[IV] The $\frac{2}{3}$ upper bound is guaranteed by Theorem 6.3.

If we choose in the beginning Δ large enough, say $\Delta \geqslant n^{\delta}$, for some constant $\delta > 0$, then we have that $\Delta \geqslant n^{\delta} \geqslant m^{\delta}$, and the negative term $-c \cdot d_1 \sqrt{m} \log(\Delta)$ is asymptotically larger, for c large enough, than the $O\left(\sqrt{m}\log(m)\right)$ term. Hence we can conclude that $S'(P', \Delta) \leqslant O\left(\left(\frac{m}{\sqrt{\Delta/3}} - \sqrt{m}\right)\log(\Delta)\right)$, which is what we wanted to prove in the beginning.

It still remains to prove that the solution holds for the boundary condition, so let Q be an instance of size $\leqslant \Delta$. Thus $S'(Q, \Delta) = 0 \leqslant c\left(\frac{|Q|}{\sqrt{\Delta/3}} - \sqrt{|Q|}\right)\log(\Delta)$, which holds if and only if $0 \leqslant \frac{|Q|}{\sqrt{\Delta/3}} - \sqrt{|Q|}$, which in turn holds if and only if $|Q| \geqslant \frac{\Delta}{3}$, but it is easy to see that this always holds. Lemma 6.2 follows entirely. ■

Now, let Λ be the number we compute with the previous algorithm. Also let c^n be the exact number of triangulations of P. By setting $\Delta = \sqrt{n}\log(n)$ we obtain that the over-counting factor of the algorithm is:

$$S(P, \Delta) = 2^{S'(P,\Delta)} = 2^{O\left(\frac{n\log(\Delta)}{\sqrt{\Delta}}\right)} = 2^{O\left(n^{\frac{3}{4}}\sqrt{\log(n)}\right)}$$

Hence $c^n \leqslant \Lambda \leqslant c^n \cdot 2^{O\left(n^{\frac{3}{4}}\sqrt{\log(n)}\right)} = c^{n+o(n)}$.

This completes the qualitative part of Theorem 6.1. It remains to discuss how much time it takes to compute Λ.

6.2.2 Running time

The running time of the algorithm can be expressed with the following recurrence:

$$T(n) < 2^{O\left(\sqrt{n}\log(n)\right)} T\left(\frac{2n}{3} + \sqrt{4n}\right)$$

Taking again $T'(n) = \log(T(n))$ yields $T'(n) := T'\left(\frac{2n}{3} + \sqrt{4n}\right) + O\left(\sqrt{n}\log(n)\right)$, which can then be solved using the well-known Akra-Bazzi Theorem for recurrences, see [52], and whose solution gives $T'(n) = O\left(\sqrt{n}\log(n)\right)$. There is however one detail missing, the stopping condition Δ. In order to use the Akra-Bazzi Theorem we need a boundary condition of $T(n) = 1$ for $1 \leqslant n \leqslant n_0$, but in the algorithm we stop the recursion whenever a sub-problem Q is of size $\leqslant \Delta$, at that point we compute the exact number

of triangulations of Q, this gives $T(|Q|) = c^{O(|Q|)} \leqslant c^{O(\Delta)}$, for some constant c whose value is not really relevant at this point. Hence the exponent in the running time of the algorithm is upper-bounded by the solution of $T'(n)$, as given by the Akra-Bazzi Theorem, plus a factor of $O(\Delta)$, *i.e.*, $T(n) = 2^{O\left(\sqrt{n}\log(n)+\Delta\right)}$. If as before we assume that $\Delta = \sqrt{n}\log(n)$ then we end up having $T(n) = 2^{O\left(\sqrt{n}\log(n)\right)}$, which concludes the proof of Theorem 6.1.

As a final remark observe that we could have used other values for Δ, rather than $\sqrt{n}\log(n)$, without violating any argument in the proofs, but then, although the quality of the approximation would have been better, the running time would have been slower. Since we see no way of not having over-counting with this algorithm, we regard $\Delta = \sqrt{n}\log(n)$ as a good trade-off.

6.3 The hardness result

In this section we show a hardness result related to our counting algorithms based on sn-paths and triangular paths, shown in Chapter 5. Observe that those algorithms are parameterized by the number k of onion layers of P. They have running times of the sort $n^{O(k)}$, thus, from the complexity point of view, it is natural to ask whether algorithms with running times of the sort $g(k) \cdot n^{O(1)}$, for some function g independent of n, are possible. That would mean that our problems belong to the FPT complexity class. Unfortunately, our techniques are general enough to solve harder problems, such as the Restricted-Triangulation-Counting-Problem explained before, on page 106.

Here we prove Theorem 6.2, which states that the Restricted-Triangulation-Counting-Problem, RTCP for short, is W[2]-hard if the parameter is considered to be the number of onion layers of P. More, this result even holds for the problem of just deciding the *existence* of a restricted triangulation.

The algorithms of Chapter 5 require little to no modification to be run on instances of RTCP, that is, those algorithms are quite generic. Since the separation FPT $\neq$ W[2] is widely believed, and we do not really know about the complexity of the counting problems studied in Chapters 4 and 5, we can still hope that by exploiting structural properties we could obtain fixed-parameter tractable algorithms for the problems studied in the two aforementioned chapters. The book by J. Flum and M. Grohe, see [37], is an excellent reference for Parameterized Complexity Theory.

6.3.1 Preliminaries

Let P be a set of n points with k onion layers, and let E be some set of pre-specified edges spanned by P. We say that a triangulation T of P is *restricted* w.r.t. E if $T \subseteq E$. Here we consider the following Restricted-Triangulation-Existence-Problem: On input (P, E), decide whether there exists a triangulation of P that is restricted w.r.t. E. This defines the Restricted-Triangulation-Counting-Problem in the natural way, and the existence problem is trivially reducible to the counting problem.

The Restricted-Triangulation-Existence-Problem, RTEP for short, was proven to be NP-complete in [53, 74]. Something very important can be observed here, namely, both reductions are actually parsimonious[v], implying #P-completeness of its natural *counting* problem, RTCP.

So far all reductions involving restricted triangulations rely heavily on the ability to specify a particular set E as part of the input. If E is instead fixed to the set of *all* edges spanned by P, we obtain the problem of counting *all* triangulations of P, which we strongly believe to be #P-complete.

In this section we parameterize RTCP and RTEP by k, the number of onion layers of P. As we mentioned before, the counting algorithm, for triangulations, presented in Section 5.3 of Chapter 5 can easily be adapted to solve RTCP, and thus also RTEP, in time $n^{O(k)}$. Our proof is by reduction from the Parameterized-Hitting-Set-Problem, PHSP for short, which is proven to be W[2]-hard in [37]. An instance A of this problem is formed by numbers $n, m, k \in \mathbb{N}$, along with sets $S_1, \ldots, S_m \subseteq [n]$, where k is considered a parameter, and $[n] := \{0, \ldots, n-1\}$. The output to A is "yes" iff there is a set $H \subseteq [n]$ of size at most k, such that $H \cap S_i \neq \emptyset$ for every $1 \leqslant i \leqslant m$.

In our reduction, several gadgets are used to transform an instance A of the hitting set problem to an instance $G_A = (P, E)$ of the Restricted-Triangulation-Existence-Problem. The reduction is an *fpt-reduction* in the sense of [37], that is, it maps every instance A with parameter k to an instance G_A with $O(k)$ onion layers. Each gadget is given by a set of points with $O(1)$ onion layers, along with a set of pre-specified edges. The gadgets that will be used are called: *pipes*, *wires*, *ORs*, *terminals*, *testers*, and *crossings*, their specifications will be given later on, for now we would like to explain how the gadgets fit together as well as the intuition behind it.

6.3.2 Construction and intuition

Given an instance A of PHSP, as explained above, we will create in polynomial time an instance $G_A = (P, E)$ of RTEP of size $\text{poly}(n, m)$ that has $O(k)$ onion layers and admits

[v] This means that there is a one-to-one correspondence between the solution sets.

a triangulation w.r.t. E iff A admits a hitting set of size $\leqslant k$. The mapping $A \mapsto G_A$ will clearly be polynomial-time computable, and thus an fpt-reduction. Figure 6.1 is a reference for the construction that follows.

In the construction, we start with parallel pipes $Q_1, \ldots, Q_k$ of n states each, and of length polynomial in m and n. Pipe Q_i lies above pipe Q_{i+1}. Let Q_i be a pipe, $1 \leqslant i \leqslant k$, and let $S_j = \{s_{j,1}, \ldots s_{j,t}\} \subseteq [n]$ be a set of instance A. We define the *stripe* $B_{i,j}$ as a set of t testers attached to Q_i that check if Q_i carries any of the values of set S_j, see Figure 6.1. The stripe $B_{i+1,j}$ will lie in the same vertical slab as $B_{i,j}$. The testers of $B_{i,j}$ are connected to a chain of or-gadgets that lies between pipes Q_i, Q_{i+1}. For $i < k$, the output of the last or-gadget in $B_{i,j}$ is carried to $B_{i+1,j}$ by a crossing-gadget, see Figure 6.1. For $i = k$, the last or-gadget in $B_{i,j}$ is connected to a terminal-gadget.

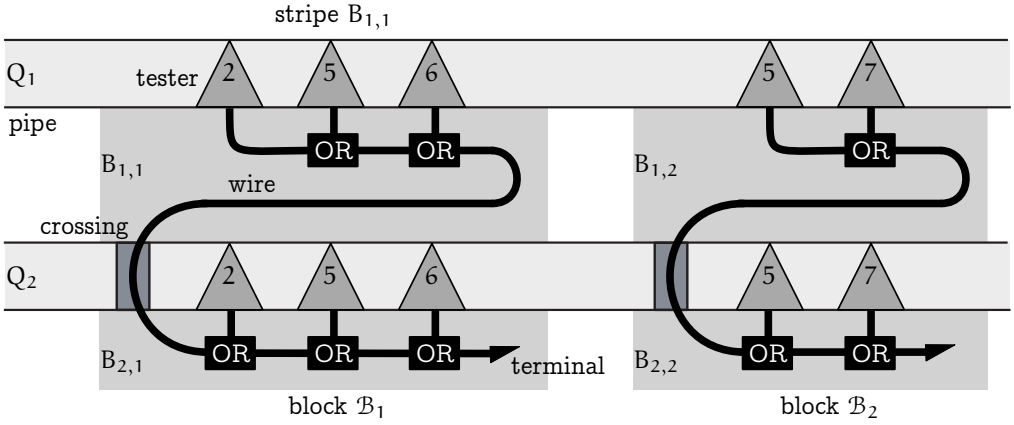

Figure 6.1 – Instance G_A produced from instance A of the PARAMETERIZED-HITTING-SET-PROBLEM with $n = 8, m = 2, k = 2$ and $S_1 = \{2,5,6\}$, $S_2 = \{5,7\}$.

The *block* $\mathcal{B}_j$ is the union of the stripes $B_{1,j}, \ldots, B_{k,j}$. The blocks $\mathcal{B}_1, \ldots, \mathcal{B}_m$ are arranged horizontally in such a way that the points in stripes $B_{i,1}, \ldots, B_{i,m}$, with $1 \leqslant i \leqslant k$, are horizontally collinear, that is, they are aligned by their y-coordinate.

Finally, P is defined to be the set of points of all the gadgets involved. To define the set E of pre-specified edges, we first include the edges of all gadgets involved. Then, the empty spaces between gadgets are triangulated arbitrarily, and these edges are added to E. We now set $G_A = (P, E)$.

The intuition behind the construction is the following: Horizontally, pipe Q_i transmits a single value between 1 and n. The testers in stripe $B_{i,j}$ verify if the value transmitted by Q_i hits one of the elements of the set S_j of A. If so, this information is transmitted vertically along block $\mathcal{B}_j$, in such a case the transmitted value is true. For this transmission we need ORs, wires and crossing gadgets. At the end of block B_j the terminal

gadget can be triangulated iff the value transmitted to it is true. If S_j is not hit by the value transmitted in Q_i, then the testers will transmit false and this value will be transmitted vertically along $\mathcal{B}_j$ until it is possibly flipped by another pipe Q_r, with $i < r \leqslant k$, thus S_j is not hit by Q_i but it is hit by Q_r. If the value transmitted to the terminal gadget in block $\mathcal{B}_j$ is false, this means that the terminal cannot be triangulated, thus no restricted triangulation of G_A exists. This in turns implies that S_j was not hit by any value transmitted by the pipes $Q_1, \ldots, Q_k$. If this is always the case then no hitting set of size at most k exists for A.

All this will be formally proven later on, for now we believe that this rough intuition is enough. Therefore we will jump now to define the gadgets formally.

6.3.3 Defining the gadgets

The basic gadget is the *pipe*, shown in Figure 6.2, whose definition is the following:

Definition 6.1 (Pipe). A *pipe* Q with n states and length $l > 4(n-1)$ consists of points $p_1 \ldots p_l$, $q_1 \ldots q_l$ with $p_t = (t,0)$, $q_t = (t,1)$, $1 \leqslant t \leqslant l$, and a set $E_Q = S \cup F \cup L_0 \cup \cdots \cup L_{n-1}$ of pre-specified edges. The individual sets that form E_Q are defined as follows:

For $1 \leqslant i \leqslant n-1$ and $1 \leqslant t \leqslant l-4i$ we define the *zig-edges* $a_{i,t} = \{p_t, q_{t+4i}\}$ and the *zag-edges* $b_{i,t} = \{q_{t+4i}, p_{t+1}\}$. For $i = 0$, we define other zig- and zag-edges by $a_{0,t} = \{p_t, q_{t+1}\}$ and $b_{0,t} = \{q_t, p_t\}$, where this time $1 \leqslant t \leqslant l-1$. For $i \in [n]$, we define the *zig-zag* $L_i = \{a_{i,1}, \ldots, a_{i,l-w}, b_{i,1}, \ldots, b_{i,l-w}\}$ with $w = 1$ for $i = 0$, and $w = 4i$ otherwise.

Next, we add the set of *completion edges* S:

$$S = \{\{p_1, q_t\} \mid 1 \leqslant t \leqslant 4(n-1)\} \cup \{\{p_{l-t}, q_l\} \mid 0 \leqslant t \leqslant 4(n-1)+2\}$$

Finally, we add the *frame edges* $F = \{\{p_i, p_{i+1}\} \mid i < l\} \cup \{\{q_i, q_{i+1}\} \mid i < l\}$.

It is clear that *any* triangulation T of a pipe Q contains exactly one zig-zag L_i, for $i \in [n]$, since different zig-zags lines cross. The sets S, F help to complete a triangulation of Q whenever zig-zag L_i is present. If $L_i \subseteq T$, we say that Q "carries" the value i in T. Note that $F \subseteq T$ holds for every triangulation T of Q. We cannot say the same about S however.

A pipe with n states will always be "horizontal", *i.e.*, it will not turn in any other direction. This is required for the final set of points to feature a bounded number of onion layers.

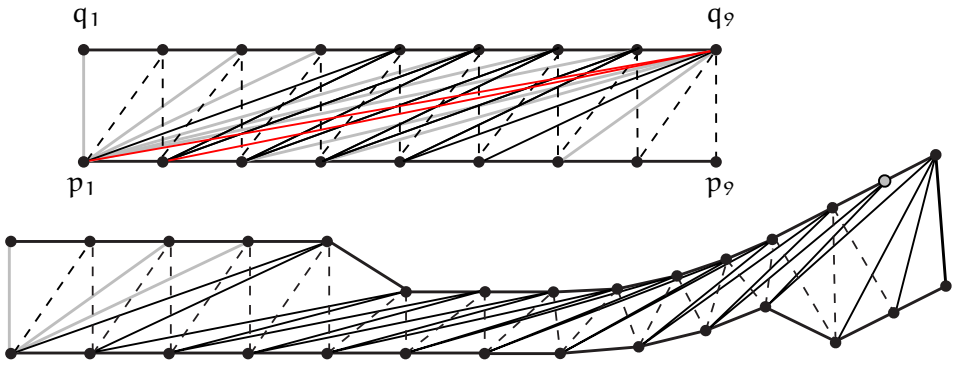

Figure 6.2 – (Top) A pipe with 3 states and $l = 9$. Thick black edges constitute F, thick gray edges constitute S, red edges are L_2, solid thin black edges are zig-zag L_1, dashed edges are zig-zag L_0. (Bottom) A stretched and bent wire with a terminal gadget attached to it.

In our construction we will also require vertical connections between pipes. These are obtained by *wires*, which are pipes with two states. Since they feature only two states, wires can be stretched by arbitrary factors, and bent by arbitrary angles, while increasing their length only by a constant additive term. This is shown in Fig. 6.2. For wires, we relabel the values 0 and 1 by false and true respectively.

The remaining gadgets for our reduction are specified and defined as follows:

Or-gadget. This gadget is connected to two input wires W_1, W_2, and to an output wire W_3, as shown in Figure 6.3. We have that: (1) If one of W_1 or W_2 carries true in some restricted[VI] triangulation T of the gadget, then W_3 may carry true. (2) If W_3 carries true in T, then at least one of W_1 or W_2 must necessarily carry true.

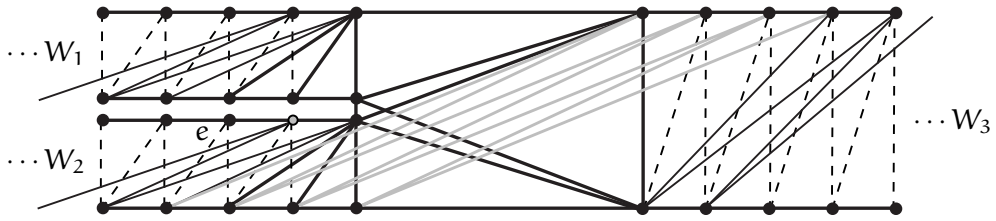

Figure 6.3 – The or-gadget. The gray edges from W_2 to W_3 are "transfer edges". An analogous set of edges is also present from W_1 to W_3, but suppressed in this figure to improve legibility.

[VI] Restricted w.r.t. the shown adjacencies.

A terminal-gadget. This gadget can be attachted to a wire W, replacing its "end part" as exemplified in the bottom part of Figure 6.2. It admits a triangulation iff W carries true.

A tester-gadget. This gadget is connected to a pipe Q, for value i at position t, between $a_{i,t}$ and $b_{i,t}$, and has an output wire W, see to the left in Figure 6.4. We have that: (1) If Q carries i in some restricted triangulation T of Q, then W may carry true. (2) If W carries true, then Q must carry i in T.

A crossing-gadget. This is a more intricate gadget which allows an input wire V to intersect a pipe Q, leaving it as an output wire W. The value carried by Q is not influenced by V. We have that: (1) If V carries true in some restricted triangulation T of the gadget, then W may carry true. (2) If W carries true, then V must necessarily carry true.

As shown in the middle in Figure 6.4, V enters the crossing-gadget from the top. If V intersects Q between points q_t and q_{t+1} then a new point r collinear with those two points is added to Q. Wire V will now enter Q between r and q_{t+1} instead, as shown in the middle in Figure 6.4. Let us assume that Q is an n-state pipe, and consider the set S formed by the points p_u such that $a_{i,u}$ is a zig-edge, of zig-zag L_i, adjacent to q_{t+1}, with $0 \leqslant i \leqslant n-1$. By definition of $a_{i,u}$ we have that $u = t - 4i + 1$ for $1 \leqslant i \leqslant n-1$, and $u = t$ for $i = 0$. There will be an output wire W_i, for zig-zag L_i, which will go out from Q between $p_u \in S$ and p_{u+1}.

Since pipes and wires are purely combinatorial objects, we have some freedom to move their points without affecting the adjacencies between the p's and q's, and without losing collinearities. Thus we will move all the p points of Q from $p_{t-4(n-1)+1}$ to p_t to the right, and condense them in such a way that we keep their linear order, thus we also keep the planarity of the zig-zags L_i, $0 \leqslant i \leqslant n-1$. The condensing part is also done in such a way that the following empty convex quadrilateral C^i_u for zig-zag L_i at $p_u \in S$ exists: Both diagonals of C^i_u have negative slopes. One diagonal of C^i_u is formed by r and p_{u+1}. The other diagonal of C^i_u is form by the point α_u of V, which is vertically aligned with r and lies three points behind r on V, and the point β_u of W_i which is vertically aligned with p_{u+1} and lies three points ahead of p_{u+1} on W_i. Points α_u and β_u, for $u = t-3$, can be seen in the middle and to the right in Figure 6.4. Observe that this re-arrangement of elements is always possible.

Now, the zig-edge $a_{i,u}$ of L_i adjacent to q_{t+1} is replaced by the edges $a'_{i,u} = \{p_u, r\}$ and $\{p_{u+1}, r\}$. The latter edge is a diagonal of C^i_u and is shown in red in Figure 6.4. The rest of the adjacencies of L_i remains the same.

Intuitively speaking, the red edge $\{p_{u+1}, r\}$ will help V to transmit false to W_i, as seen to the right in Figure 6.4 for $i = 1$. Thus we also need to add the edges

that will help V to transmit true to W_i. Those edges are shown on solid black for $i = 1$ to the right in Figure 6.4. The adjacencies are equivalent for any other $0 \leqslant i \leqslant n-1$. Observe that all these adjacencies intersect neither $a'_{i,u}$ nor $b_{i,u}$.

Finally, the output wires $W_0, \ldots, W_{n-1}$ are connected to a chain of or-gadgets, as shown in the middle in Figure 6.4, whose output is precisely the output wire W.

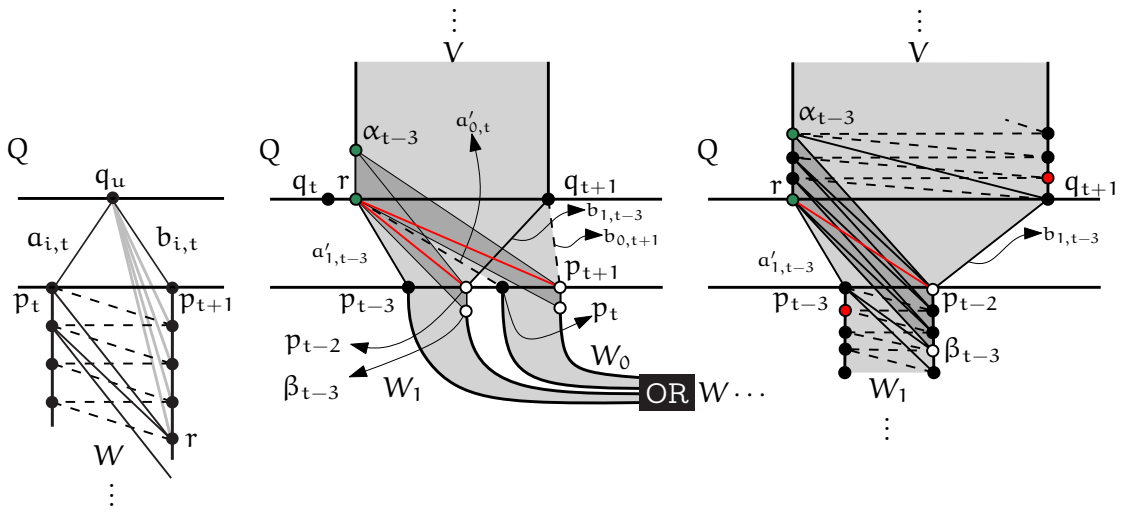

Figure 6.4 – To the left the tester-gadget for i at t. Q is modified by shifting, for $k > 0$, all p_{t+k} and q_{u+k} to the right until the triangle r, p_{t+1}, q_u is oriented counter-clockwise. In the middle a crossing between pipe Q and input wire V which becomes output wire W. To the right the details of the crossing for $i = 1$ at p_{t-3}.

6.3.4 Formal proofs

Lemma 6.3. *All the previous gadgets fulfill their specifications.*

Proof. **Or-gadget:** (1) Assume without loss of generality that W_2 carries true in some restricted triangulation T of the gadget. Observe that W_3 carries true or false in T depending on whether the transfer edges, shown in gray in Figure 6.3, are chosen. For (2) note that if W_3 carries true in T, the transfer edges from either W_1 or W_2, say W_2 without loss of generality, must be present. If W_2 carried false, it can do it only up to edge e shown in Figure 6.3, since all following edges intersect transfer edges. But then, the gray point fails to be part of a triangle in *any* restricted triangulation of the gadget, thus W_2 must necessarily carry true in T.

Terminal: If W carries true, the terminal is triangulated as shown at the bottom of Figure 6.2. However, if W carries false, then the gray point shown in the same figure fails to be in a triangle of the restricted triangulation of the gadget.

Tester: For (1) assume Q carries i in some restricted triangulation T. Since no gray edges in the tester intersect L_i, they can be added to T or not. That would make W carry true or false respectively. For (2) given some restricted triangulation T, in which W carries true, all gray edges must be present in T. But the gadget is designed such that for every $0 \leqslant j \neq i \leqslant n-1$, there is an edge $e \in L_j$ of Q that intersects both $a_{i,t}$ and $b_{i,t}$, and thus all gray edges. Therefore $e \notin T$, and hence $L_j \nsubseteq T$, forcing $L_i \subseteq T$.

Crossing: (1) Let T be a restricted triangulation of the whole gadget in which Q carries i. Observe that if either, V or W_i, carries true in T, then the black solid edges that cross Q from V to W_i, shown to the right in Figure 6.4 for $i = 1$, must be present. Those edges in turn imply that the other gadget must necessarily carry true in T as well, otherwise the red points shown to the right in Figure 6.4 will fail to be part of T, which would give us a contradiction since T is a triangulation. Thus V carries true iff W_i carries true, as long as Q carries i. By using the chain of or-gadgets that the W_j's are connected to we could leave the output wire W carrying true. (2) Assume that W carries true and Q carries i in T. Observe that in the chain of or-gadgets that the W_j's are connected to, we can always force to transmit true from W to W_i, while we transmit false to every other W_j, $0 \leqslant j \neq i \leqslant n-1$. This in turn will force the edges that cross Q from W_i to V to be included in T, the black solid edges that cross Q from W_i to V shown to the right in Figure 6.4 for $i = 1$. This will make V carry true.

A triangulation is also possible if V and W carry false. If Q carries i in T, then the edges $a'_{i,u} = \{p_u, r\}$ and $\{p_{u+1}, r\}$, for some $t - 4(n-1) + 1 \leqslant u \leqslant t$, are also present in T. Thus we transmit false from W to every W_j, $0 \leqslant j \leqslant n-1$. However, as we said before, the red edge $\{p_{u+1}, r\}$ will help to transmit false from W_i to V through Q. Thus V would also carry false in T. ■

Theorem 6.2 follows from the following lemma:

Lemma 6.4. *G_A has $O(k)$ onion layers and admits a triangulation iff A admits a hitting set of size $\leqslant k$.*

Proof. Consider the number of different y-coordinates of P. This is an upper bound for the number of onion layers of P. The pipes contribute $2k$ different y-coordinates. Every other gadget features $O(1)$ different y-coordinates. Each wire can be stretched and bent with $O(1)$ overhead, thus giving $O(1)$ different y-coordinates. Since the points in stripes

$B_{i,1}, \ldots, B_{i,m}$ are aligned by their y-coordinates, each set $B_{i,1} \cup \cdots \cup B_{i,m}$ has $O(1)$ different y-coordinates. This totals to $2k + O(k) = O(k)$ different y-coordinates among all points in P.

Given a hitting set $H = \{x_1, \ldots, x_k\}$ of k elements, we construct a triangulation that uses only edges from E as follows: For every $i \leqslant k$, make Q_i carry x_i. For every $j \leqslant m$ pick some $x = x_i \in H$ such that $x \in S_j$. In stripe $B_{i,j}$ triangulate the output wire of the tester for x to carry true, and transmit this true value along the or-gadgets of $B_{i,j}$. When crossing a pipe Q_z, with $z > i$, the true value will get transmitted through the output wire W_z of the corresponding crossing-gadget. The true value will eventually reach the terminal of $\mathcal{B}_j$, which can then be triangulated without problems.

On the other hand, the values $H = \{x_1, \ldots, x_k\}$ carried by the pipes $Q_1, \ldots, Q_k$ in any restricted triangulation of G_A form a hitting set. To see this, observe that every terminal must be triangulated, so the wire of every block $\mathcal{B}_j$ must carry true at some place. Thus, the output of some or-gadget in $\mathcal{B}_j$ must carry true. Consider the first or-gadget that fulfills this top-down, and say it lies in stripe $B_{i,j}$. This or-gadget must be connected to a tester that outputs true. This implies $x_i \in S_j$ and $H \cap S_j \neq \emptyset$. ■

6.4 Experimental results on counting triangulations

We have implemented the sn-path algorithm presented in § 5.3 of Chapter 5 and we compare it with the algorithm presented in [70], and also with our own T-path algorithm presented in § 4.2 of Chapter 4. The implementations for the latter two algorithms were kindly provided by Saurabh Ray. All experiments were run on a server generously provided by Prof. Bernd Finkbeiner, head of the Reactive Systems group at Saarland University. All implementations are single-threaded, so all algorithms were run a on single core of a dual-core processor AMD Opteron at 2.6 Ghz. Linux was the used operating system, and the amount of RAM available was 122 GB. Finally, all implementations use the GMP library to handle big numbers. All statistics here reported were obtained from the output of the command 'time -v'.

The main idea behind the experiments was to obtain evidence of the practical limits of the algorithms, thus they are really provided without statistical analysis. If we denote the number of points by n, the number of onion layers by k, and the size of the convex hull by h, we were interested in knowing for different values of those parameters what are the largest sets of points we can solve. Also, besides providing the number of triangulations, we also provide: (1) Total number of sub-problems generated by each algorithm, (2) Memory consumption, and (3) Total running time. Since we used memoization in all algorithms, the total number of sub-problems is just the size of the database at the end

of execution with exception of the T-path algorithm, there we rather kept the largest number of T-paths the algorithm encountered during its execution.

We have four kinds of sets of points, of selected cardinalities, we ran the algorithms on: (1) Sets of points having three onion layers. (2) Sets of points generated in a square (3) Sets of points having the largest possible number of onion layers, w.r.t. the cardinality of the set (4) Grids. Sets (1), (2) and (3) were generated at random. For (1) we generated random points on three concentric circles and we only kept configurations having three onion layers.

Table 6.1 summarizes the largest sets of points, of each type, that we were able to solve within 140 hours, the complete results can be seen in the tables at the end of the chapter. Results for (1) are shown in Tables 6.2 and 6.3, for (2) in Tables 6.4 and 6.5, for (3) in Tables 6.6 and 6.7, and for (4) in Tables 6.8 and 6.9. In the tables the algorithm of [70] is called "ray-seidel", and our algorithms simply "sn-paths" and "t-paths". We could not run ray-seidel on sets of kind (4) due to degeneracy. All columns are self-explanatory except for the columns "Base" and "Exp". The former refers to the base c, truncated to two decimal digits, of a number expressed as c^n. The latter refers to the term d, also truncated to two decimal digits, of a number expressed as $n^{d \cdot k}$, this makes sense for sn-paths since we know that for fixed k the running time is $n^{O(k)}$.

	# Points			
	(1)	(2)	(3)	(4)
ray-seidel	43	43	34	NA
sn-paths	80	43	28	6x17
t-paths	30	33	28	6x7

Table 6.1 – Largest sets of points of kind (i), $1 \leqslant i \leqslant 4$, solved by each algorithm within 140 hours.

Since we are interested in the largest n we can solve, we started the experiments with at least 25 points, below this threshold all algorithms perform very well, where ray-seidel is notably the fastest, giving answers in at most a couple of seconds, and sn-paths the slowest for $k = 8$. All empty entries, except for the last entry of sn-paths in Tables 6.4 and 6.5, mean that the corresponding algorithm consumed all available RAM memory *before* finishing the corresponding set of points. In the same sense, one complete empty row means that *no* algorithm managed to finish the corresponding set of points. The last entry of sn-paths in tables 6.4 and 6.5 was explicitly stopped due to its potentially large running time.

To verify the correctness of the algorithms we ran them on configurations available in [1, 47], for which an answer is known via other algorithms. We also run them on sets of points in convex position, there the number of triangulations is a Catalan number. In all cases the three algorithms confirmed the known answers.

The experiments turned out to be what we had expected, namely, generally worse behavior as the number of onion layers increases, since the number of triangulations should certainly increase with the number of onion layers. The experiments show however that *all* algorithms are counting triangulations by generating far fewer sub-problems: Having a glimpse at the number of sub-problems in Tables 6.2 to 6.9, each looks as something of the sort $\sqrt{|\mathcal{F}_T(P)|}$, which was already reported in [70] for the ray-seidel algorithm.

The t-paths and ray-seidel algorithms showed a consistent behavior across all experiments, the former being notably the most expensive overall computationally speaking. This came at first as a surprise since t-paths is a very simple algorithm, and it has a running time linear in the number of T-paths it encounters, but on a second thought one realizes that the number of T-paths is expected to be *always* exponential, thus there is no doubt that they are really the bottleneck of the algorithm. The ray-seidel algorithm lived up to its expectations, it turned out to be simply the fastest algorithm, but this came with the price of being *very* resource-consuming. We can see in the tables that the resources the algorithm uses increase very fast, at that rate we could say that increasing RAM to a couple of Terabytes will not really allow us to run the algorithm on significantly larger set of points. However, there are other techniques we could use to alleviate this situation, we could for example decide to store only a subset of the produced sub-problems and re-compute a sub-problem whenever needed. Since apparently computing sub-problems is very fast, we could expect that this method does not severely blow up the running time.

Now turning to sn-paths, the algorithm really performed best for three convex layers, see Tables 6.2 and 6.3. For those configurations the algorithm allowed us to go up to 80 points in a "reasonable" amount of time without exhausting the RAM, which is almost twice as much as the ray-seidel algorithm allowed. In this regard we believe that increasing computational power and resources, say 512 GB of RAM, could allow us to go somewhere near 160 points, the other two algorithms would get nowhere close to this number of points *per se*. This "nice" behavior can also be seen in Tables 6.8 and 6.9 with grids having three and four onion layers, however, grids are believed to have far less triangulations than sets of points in general position, this is also supported by the experiments. For fewer than three onion layers the algorithm gets better, so they are really not an issue. Also, since the running time of sn-paths can be expressed as $n^{d \cdot k}$, for some positive d, the idea of column Exp in the tables was to see whether that value comes out roughly as a constant for the same values of k, however, the data set seems to be small to show this. We believe that the right value should be $3 \leqslant d \leqslant 4$. Finally,

by increasing the number of onion layers, Tables 6.4 to 6.7, we can see how the behavior of sn-paths quickly deteriorates, in all aspects, up to the point of being comparable to t-paths for $k = \lceil \frac{n}{3} \rceil$, where sometimes the latter is even faster.

The conclusions of the experiments really suggest themselves. In the "low" end, up to 25 points, any algorithm will do but ray-seidel is the fastest. In the "high" end it really seems that sn-paths is a better choice due to its smaller memory footprint, so up to 6-7 onion layers we would stick with it, but beyond that number of onion layers we would consider ray-seidel a better option.

6.5 Conclusions

In this chapter we have shown two results related to the problem of counting triangulations. The first result shown was an algorithm to compute the number of triangulations approximately. Although this algorithm fails to give an exact answer, it correctly computes the base c of the number of triangulations c^n, and it does so in sub-exponential time. No algorithm with this properties was known before.

The second result shown was a hardness result of a very particular instance of the problem of counting triangulations exactly, namely the Restricted-Triangulation-Counting-Problem (RTCP). We showed that this problem is W[2]-hard if the parameter is the number of onion layers of the set of points it is defined on. The algorithm for counting triangulations shown in Chapter 5 needs little to no modifications to be run on instances of RTCP, and the separation FPT $\neq$ W[2] is widely believed, so we can still hope that, by exploiting structural properties of triangulations, and also of other crossing-free structures, we can obtain FPT algorithms for the counting problems studied in Chapters 4 and 5. Thus, one interesting question at this moment is: Do those counting problems belong to FPT or not?

Finally, we showed experimental results comparing the algorithms for counting triangulations of Chapters 4 and 5, and the algorithm of [70]. Those experiments give a rough idea of what to expect when running each one of those algorithms on real configurations of points. It would be very interesting to see a hybrid algorithm that uses the sn-path and the ray-seidel algorithms, if possible. That algorithm could combine the small memory footprint of the sn-path algorithm with the fast execution of the ray-seidel algorithm. This could allow us to solve larger sets of points.

				Time in hh:mm:ss.ms			RAM in Mb		
n	h	#Triangulations	Base	ray-seidel	sn-paths	t-paths	ray-seidel	sn-paths	t-paths
30	10	161014656152655441	≈ 3.74	20.18	55.77	1:21:19	303	33	39740
	10	312513373686594183	≈ 3.82	1:07.72	1:09.17	3:20:58	1160	38	87555
33	11	32155601714553665796	≈ 3.90	3:50.96	2:19.19		2762	62	
	11	68598010833407738067	≈ 3.99	58.43	2:30.94		857	62	
37	12	9334947679230323509429	≈ 3.92	1:53.60	3:43.51		1521	90	
	12	31113068813012076443512	≈ 4.05	3:20.41	4:42.24		2465	97	
40	14	2642143054680217856074126	≈ 4.07	18:12.16	8:10.26		12557	131	
	15	2903778262295075928823011	≈ 4.08	7:01.40	9:50.98		4325	149	
43	14	45237169780816239658305 5656	≈ 4.16	1:34:48	21:39.66		53263	243	
	14	46155021476436988101856 4051	≈ 4.16		19:25.53			242	
47	16	15775971054067198543662 1922639	≈ 4.18		32:46.68			363	
	15	34103758523867834671037 2748758	≈ 4.24		39:33.42			420	
50	16	54782168649020627430413 001433261	≈ 4.31		1:06:53			606	
	16	15899759272368397775850 1079915910	≈ 4.40		1:07:19			553	
60	21	38305193272256676568359 1748023039 0004428	≈ 4.56		7:51:53			2006	
	20	19003078026633792670003 3771493586 96361338	≈ 4.69		8:59:36			2063	
70	25	48142357864275890897765 1277967532 53695164862078	≈ 4.64		22:59:43			3937	
	23	22441154772967246982370 9078962020 667530864087588	≈ 4.74		26:54:49			4506	
80	26	39697885166896605395758 2788796899 340386475542246 4030090	≈ 4.81		81:29:23			8932	
	26	18527998227718271520712 6583259662 533934854740404 52858832	≈ 4.90		90:34:22			9617	

Table 6.2 — n random points on a square, having three onion layers and h points on their convex hull.

		# Sub-problems						
n	h	ray-seidel	Base	sn-paths	Base	Exp	t-paths	Base
30	10	2050514	≈ 1.62	215732	≈ 1.50	≈ 1.20	147633229	≈ 1.87
	10	7879754	≈ 1.69	246657	≈ 1.51	≈ 1.21	351513627	≈ 1.92
33	11	18992928	≈ 1.66	405580	≈ 1.47	≈ 1.23		
	11	5812991	≈ 1.60	410357	≈ 1.47	≈ 1.23		
37	12	10027300	≈ 1.54	575255	≈ 1.43	≈ 1.22		
	12	16250100	≈ 1.56	626274	≈ 1.43	≈ 1.23		
40	14	82635240	≈ 1.57	866278	≈ 1.40	≈ 1.23		
	15	28333612	≈ 1.53	982791	≈ 1.41	≈ 1.24		
43	14	347603518	≈ 1.57	1604269	≈ 1.39	≈ 1.26		
	14			1591423	≈ 1.39	≈ 1.26		
47	16			2287764	≈ 1.36	≈ 1.26		
	15			2720786	≈ 1.37	≈ 1.28		
50	16			3631525	≈ 1.35	≈ 1.28		
	16			3998798	≈ 1.35	≈ 1.29		
60	21			12527119	≈ 1.31	≈ 1.33		
	20			13076694	≈ 1.31	≈ 1.33		
70	25			23762305	≈ 1.27	≈ 1.33		
	23			27937551	≈ 1.27	≈ 1.34		
80	26			54047260	≈ 1.24	≈ 1.35		
	26			58561612	≈ 1.25	≈ 1.36		

Table 6.3 – Number of sub-problems generated by the configurations (entry-wise) presented in Table 6.2.

					Time in hhh:mm:ss.ms			RAM in Mb		
n	k	h	#Triangulations	Base	ray-seidel	sn-paths	t-paths	ray-seidel	sn-paths	t-paths
30	5	9	29762284427845618	≈ 3.54	7.60	3:56.61	16:37.83	130	141	7063
	6	7	54648952555202115	≈ 3.61	30.69	16:17.12	21:19.74	470	535	9955
33	5	11	8830953374442248378	≈ 3.75	34.80	14:47.26	4:09:15	643	394	63051
	6	7	23407918365649149382	≈ 3.86	15.10	1:10:34	5:03:29	288	1292	82817
37	5	11	8317197892568798832050	≈ 3.91	3:35.27	1:15:00		2796	1524	
	5	13	15347609782987966767248	≈ 3.97	15:23.53	2:16:32		10707	1957	
40	6	12	1146138971033715203926926	≈ 3.99	25:42.43	13:29:43		18525	8889	
	7	10	5050493282169462429012536	≈ 4.14	1:35:45	46:49:41		54128	25533	
43	6	10	98140331329825983429220292 5	≈ 4.24	3:20:54	107:48:48		116506	37407	
	7	8								

Table 6.4 — n random points on a square, having k onion layers and h points of their convex hull.

			# Sub-problems						
n	k	h	ray-seidel	Base	sn-paths	Base	Exp	t-paths	Base
30	5	9	854579	≈ 1.56	947262	≈ 1.58	≈ 0.80	23535563	≈ 1.76
	6	7	3150228	≈ 1.64	3590878	≈ 1.65	≈ 0.73	35951972	≈ 1.78
33	5	11	4245399	≈ 1.58	2554063	≈ 1.56	≈ 0.84	216691338	≈ 1.78
	6	7	1907449	≈ 1.54	8731943	≈ 1.62	≈ 0.76	283621585	≈ 1.80
37	5	11	18477670	≈ 1.57	9735430	≈ 1.54	≈ 0.89		
	5	13	70483691	≈ 1.62	12535632	≈ 1.55	≈ 0.90		
40	6	12	121049523	≈ 1.59	56587195	≈ 1.56	≈ 0.80		
	7	10	354717051	≈ 1.63	155716531	≈ 1.60	≈ 0.73		
43	6	10	752596823	≈ 1.60	239084256	≈ 1.56	≈ 0.85		
	7	8							

Table 6.5 — Number of sub-problems generated by the configurations (entry-wise) presented in Table 6.4.

				Time in hh:mm:ss.ms			RAM in Mb		
k	n	#Triangulations	Base	ray-seidel	sn-paths	t-paths	ray-seidel	sn-paths	t-paths
9	25	248441701550196	≈ 3.76	7.82	1:36:07	6:35.48	154	3828	3801
	27	6632755933105064	≈ 3.85	1:16.84	9:56:09	26:15.48	1239	17406	13564
10	28	134806114688321888	≈ 4.09	1:04.33	39:29:17	37:51.20	1130	56197	14147
	28	259051751512786147	≈ 4.18	58.55	32:31:58	6:33:56	903	41745	122190
11	32	188748482026800154083	≈ 4.30	1:43:22			70647		
	33	2680138023948109608080	≈ 4.46	1:38:04			62477		
12	34	16605186163166445755560	≈ 4.50	2:52:16			114676		
	34								

Table 6.6 – n random points having $k = \lceil \frac{n}{3} \rceil$ onion layers.

		# Sub-problems					
k	n	ray-seidel	Base	sn-paths	Base	t-paths	Base
9	25	1078399	≈ 1.74	24811886	≈ 1.97	13146548	≈ 1.92
	27	8738535	≈ 1.80	110817524	≈ 1.98	44831603	≈ 1.92
10	28	8015023	≈ 1.76	347448787	≈ 2.01	52145375	≈ 1.88
	28	6303203	≈ 1.74	266661064	≈ 1.99	412317726	≈ 2.03
11	32	478692844	≈ 1.86				
	33	423236754	≈ 1.82				
12	34	773361622	≈ 1.82				
	34						

Table 6.7 – Number of sub-problems generated by the configurations (entry-wise) presented in Table 6.6.

				Time in hhh:mm:ss.ms			RAM in Mb	
n	m	k	#Triangulations	Base	sn-paths	t-paths	sn-paths	t-paths
6	6	3	260420548144996	≈ 2.51	10.36	1:55.74	16	1175
6	7	3	341816489625522032	≈ 2.61	45.79	22:07.25	41	8442
6	8	3	464476385680935656240	≈ 2.69	2:31.70		107	
6	9	3	645855159466371391947660	≈ 2.76	7:28.22		213	
6	10	3	913036902513499041820702784	≈ 2.81	17:27.63		460	
6	11	3	1306520849733616781789190513820	≈ 2.85	39:35.88		840	
6	12	3	1887591165891651253904039432371172	≈ 2.89	1:22:15		1555	
6	13	3	2747848427721241461905176361078147168	≈ 2.93	2:50:42		2479	
6	14	3	4024758386310801427793602374466243714608	≈ 2.96	5:14:23		4439	
6	15	3	5924744736041718687622958191829471010847132	≈ 2.98	8:43:14		6315	
6	16	3	875795619957126111669022659876450114208849686O	≈ 3.01	14:55:29		10344	
6	17	3	1299121595791657763525109561385946517621653010608O	≈ 3.03	26:08:38		15023	
7	7	4	1999206934751133055518	≈ 2.72	6:09.75		187	
7	8	4	12169409954141988707186052	≈ 2.80	36:59.53		869	
7	9	4	760833363329475136555549189O4	≈ 2.87	2:28:42		2344	
7	10	4	484772512167266688498399632918196	≈ 2.93	8:12:59		6465	
7	11	4	3131521959869770128138491287826065904	≈ 2.97	27:42:55		14870	
7	12	4	204437676119275998232172917694684494885O8	≈ 3.01	67:06:41		34752	
8	8	4	33263384084411310375159799592O	≈ 2.89	6:00:49		6171	
8	9	4	936936351750120881953042996728O708	≈ 2.96	32:49:11		19071	
8	10	4	2696211097537325182524932578284131372O2	≈ 3.02	139:58:01		75220	

Table 6.8 — Grid of n by m with k onion layers.

			# Sub-problems				
n	m	k	sn-paths	Base	Exp	t-paths	Base
6	6	3	69908	≈ 1.36	≈ 1.03	4025520	≈ 1.52
6	7	3	207193	≈ 1.33	≈ 1.09	27908087	≈ 1.50
6	8	3	465416	≈ 1.31	≈ 1.12		
6	9	3	1002029	≈ 1.29	≈ 1.15		
6	10	3	1883205	≈ 1.27	≈ 1.17		
6	11	3	3409331	≈ 1.25	≈ 1.19		
6	12	3	5705962	≈ 1.24	≈ 1.21		
6	13	3	9417222	≈ 1.22	≈ 1.22		
6	14	3	14471156	≈ 1.21	≈ 1.24		
6	15	3	22201708	≈ 1.20	≈ 1.25		
6	16	3	32491047	≈ 1.19	≈ 1.26		
6	17	3	46979052	≈ 1.18	≈ 1.27		
7	7	4	972496	≈ 1.32	≈ 0.88		
7	8	4	3527752	≈ 1.30	≈ 0.93		
7	9	4	10558836	≈ 1.29	≈ 0.97		
7	10	4	25013282	≈ 1.27	≈ 1		
7	11	4	55453561	≈ 1.26	≈ 1.02		
7	12	4	109901193	≈ 1.24	≈ 1.04		
8	8	4	14569428	≈ 1.29	≈ 0.99		
8	9	4	50333235	≈ 1.27	≈ 1.03		
8	10	4	122283519	≈ 1.26	≈ 1.06		

Table 6.9 – Number of sub-problems generated by the configurations (entry-wise) presented in Table 6.8.

LIST OF FIGURES

BIBLIOGRAPHY

[1] O. Aichholzer. Counting triangulations - olympics. `http://www.ist.tugraz.at/staff/aichholzer/research/rp/triangulations/counting/`. One citation on page 121.

[2] O. Aichholzer. The path of a triangulation. In *Symposium on Computational Geometry*, pages 14–23, 1999. 8 citations on pages 11, 12, 55, 56, 71, 88, and 104.

[3] O. Aichholzer, F. Aurenhammer, H. Krasser, and B. Speckmann. Convexity minimizes pseudo-triangulations. *Comput. Geom.*, 28(1):3–10, 2004. One citation on page 54.

[4] O. Aichholzer, T. Hackl, M. Hoffmann, A. Pilz, G. Rote, B. Speckmann, and B. Vogtenhuber. Plane graphs with parity constraints. In F. K. H. A. Dehne, M. L. Gavrilova, J.-R. Sack, and C. D. Tóth, editors, *WADS*, volume 5664 of *Lecture Notes in Computer Science*, pages 13–24. Springer, 2009. 5 citations on pages 9, 29, 30, 45, and 50.

[5] O. Aichholzer, T. Hackl, C. Huemer, F. Hurtado, and B. Vogtenhuber. Large bichromatic point sets admit empty monochromatic 4-gons. *SIAM J. Discrete Math.*, 23(4):2147–2155, 2010. One citation on page 30.

[6] O. Aichholzer, G. Rote, B. Speckmann, and I. Streinu. The zigzag path of a pseudo-triangulation. In F. K. H. A. Dehne, J.-R. Sack, and M. H. M. Smid, editors, *WADS*, volume 2748 of *Lecture Notes in Computer Science*, pages 377–388. Springer, 2003. 8 citations on pages 12, 55, 57, 58, and 75.

[7] M. Ajtai, V. Chvátal, M. Newborn, and E. Szemerédi. Crossing-free subgraphs. In P. L. Hammer, A. Rosa, G. Sabidussi, and J. Turgeon, editors, *Theory and Practice of Combinatorics A collection of articles honoring Anton Kotzig on*

the occasion of his sixtieth birthday, volume 60 of *North-Holland Mathematics Studies*, pages 9 – 12. North-Holland, 1982. One citation on page 54.

[8] V. Alvarez. Even triangulation of planar set of points with steiner points. In *EuroCG*, 2010. One citation on page 4.

[9] V. Alvarez, K. Bringmann, R. Curticapean, and S. Ray. Counting crossing-free structures. In T. K. Dey and S. Whitesides, editors, *Symposium on Computational Geometry*, pages 61–68. ACM, 2012. One citation on page 4.

[10] V. Alvarez and A. Nakamoto. Colored quadrangulations with steiner points. In *EuroCG*, 2012. One citation on page 4.

[11] V. Alvarez, T. Sakai, and J. Urrutia. Bichromatic quadrangulations with steiner points. *Graph. Comb.*, 23(1):85–98, 2007. 16 citations on pages 9, 18, 19, 23, 27, and 28.

[12] E. Anagnostou and D. G. Corneil. Polynomial-time instances of the minimum weight triangulation problem. *Comput. Geom.*, 3:247–259, 1993. One citation on page 93.

[13] T. Auer and M. Held. Heuristics for the generation of random polygons. In F. Fiala, E. Kranakis, and J.-R. Sack, editors, *CCCG*, pages 38–43. Carleton University Press, 1996. One citation on page 104.

[14] D. Avis and K. Fukuda. Reverse search for enumeration. *Discrete Applied Mathematics*, 65(1-3):21–46, 1996. One citation on page 54.

[15] S. Bereg. Enumerating pseudo-triangulations in the plane. *Comput. Geom.*, 30(3):207–222, 2005. 2 citations on pages 54 and 57.

[16] S. Bespamyatnikh. Enumerating pseudo-triangulations in the plane. In *CCCG*, pages 162–166, 2002. One citation on page 54.

[17] S. Bespamyatnikh. Computing homotopic shortest paths in the plane. *J. Algorithms*, 49(2):284–303, 2003. One citation on page 83.

[18] J.-D. Boissonnat and M. Yvinec. *Algorithmic geometry*. Cambridge University Press, 1998. One citation on page 4.

[19] N. Bonichon, C. Gavoille, and N. Hanusse. Canonical decomposition of outerplanar maps and application to enumeration, coding and generation. *J. Graph Algorithms Appl.*, 9(2):185–204, 2005. One citation on page 74.

[20] P. Bose, S. Ramaswami, G. T. Toussaint, and A. Turki. Experimental results on quadrangulations of sets of fixed points. *Computer Aided Geometric Design*, 19(7):533–552, 2002. 2 citations on pages 17 and 18.

[21] P. Bose and G. T. Toussaint. Characterizing and efficiently computing quadrangulations of planar point sets. *Computer Aided Geometric Design*, 14(8):763–785, 1997. One citation on page 17.

[22] D. Bremner, F. Hurtado, S. Ramaswami, and V. Sacristan. Small strictly convex quadrilateral meshes of point sets. *Algorithmica*, 38(2):317–339, 2003. One citation on page 17.

[23] B. Chazelle, H. Edelsbrunner, M. Grigni, L. J. Guibas, J. Hershberger, M. Sharir, and J. Snoeyink. Ray shooting in polygons using geodesic triangulations. *Algorithmica*, 12(1):54–68, 1994. One citation on page 53.

[24] L. P. Chew. Constrained delaunay triangulations. In *Symposium on Computational Geometry*, pages 215–222, 1987. One citation on page 100.

[25] C. Cortés, A. Márquez, A. Nakamoto, and J. Valenzuela. Quadrangulations and 2-colorations. In *EuroCG*, pages 65–68. Technische Universiteit Eindhoven, 2005. One citation on page 18.

[26] K. Dalal. Counting the onion. *Random Struct. Algorithms*, 24(2):155–165, 2004. One citation on page 98.

[27] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, second edition, 2000. One citation on page 4.

[28] E. Demaine. Simple polygonizations. `http://erikdemaine.org/polygonization/`. One citation on page 91.

[29] E. Demaine, J. S. B. Mitchell, and J. O'Rourke. Problem 16: Simple polygonalizations. `http://cs.smith.edu/~orourke/TOPP/P16.html#Problem.16`. 2 citations on pages 93 and 104.

[30] K. Diks, L. Kowalik, and M. Kurowski. A new 3-color criterion for planar graphs. In L. Kucera, editor, *WG*, volume 2573 of *Lecture Notes in Computer Science*, pages 138–149. Springer, 2002. 2 citations on pages 30 and 31.

[31] H. Djidjev and S. M. Venkatesan. Reduced constants for simple cycle graph separation. *Acta Inf.*, 34(3):231–243, 1997. One citation on page 107.

[32] A. Dumitrescu, B. Gärtner, S. Pedroni, and E. Welzl. Enumerating triangulation paths. *Comput. Geom.*, 20(1-2):3–12, 2001. 2 citations on pages 57 and 71.

[33] A. Dumitrescu, A. Schulz, A. Sheffer, and C. D. Tóth. Bounds on the maximum multiplicity of some common geometric graphs. In T. Schwentick and C. Dürr, editors, *STACS*, volume 9 of *LIPIcs*, pages 637–648. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011. 2 citations on pages 3 and 104.

[34] H. Edelsbrunner. *Algorithms in combinatorial geometry*. Springer-Verlag New York, Inc., New York, NY, USA, 1987. One citation on page 4.

[35] P. Epstein and J.-R. Sack. Generating triangulations at random. *ACM Trans. Model. Comput. Simul.*, 4(3):267–278, 1994. One citation on page 88.

[36] S. Fisk. A short proof of chvátal's watchman theorem. *J. Comb. Theory, Ser. B*, 24(3):374, 1978. One citation on page 33.

[37] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2006. 5 citations on pages 14, 106, 111, and 112.

[38] J. E. Goodman, J. Pach, and R. Pollack, editors. *Surveys on Discrete and Computational Geometry: Twenty Years Later*. American Mathematical Society, Providence, RI, USA, 2008. One citation on page 54.

[39] J. E. Goodman and R. Pollack. Multidimensional sorting. *SIAM J. Comput.*, 12(3):484–507, 1983. One citation on page 2.

[40] J. E. Goodman and R. Pollack. Upper bounds for configurations and polytopes in r^d. *Discrete & Computational Geometry*, 1:219–227, 1986. 2 citations on pages 2 and 54.

[41] J. E. Goodman and R. Pollack. The complexity of point configurations. *Discrete Applied Mathematics*, 31(2):167–180, 1991. One citation on page 2.

[42] M. T. Goodrich and R. Tamassia. Dynamic ray shooting and shortest paths in planar subdivisions via balanced geodesic triangulations. *J. Algorithms*, 23(1):51–73, 1997. One citation on page 53.

[43] R. L. Graham, D. E. Knuth, and O. Patashnik. *Concrete mathematics - a foundation for computer science (2. ed.)*. Addison-Wesley, 1994. One citation on page 74.

[44] P. J. Heawood. On the four-color map theorem. *Quart. J. Pure Math.*, 29:270–285, 1898. One citation on page 30.

[45] M. Heredia and J. Urrutia. On convex quadrangulations of point sets on the plane. In J. Akiyama, W. Y. C. Chen, M. Kano, X. Li, and Q. Yu, editors, *CJCDGCGT*, volume 4381 of *Lecture Notes in Computer Science*, pages 38–46. Springer, 2005. One citation on page 17.

[46] Ø. Hjelle and M. Dæhlen. *Triangulations and Applications (Mathematics and Visualization)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. One citation on page 100.

[47] V. Kaibel and G. M. Ziegler. Counting lattice triangulations. In C. D. Wensley, editor, *Surveys in Combinatorics*, volume 307 of *London Mathematical Society Lecture Note Series*, pages 277–307. Cambridge University Press, 2003. One citation on page 121.

[48] S. Kato, R. Mori, and A. Nakamoto. Quadrangulations on 3-colored point sets with steiner points and their winding numbers. *Submitted. Preliminary version in Proc. XIV Spanish Meeting on Computational Geometry*, pages 133–136, 2011. 7 citations on pages 18, 19, 20, 21, and 23.

[49] N. Katoh and S.-I. Tanigawa. Fast enumeration algorithms for non-crossing geometric graphs. *Discrete & Computational Geometry*, 42(3):443–468, 2009. One citation on page 54.

[50] H. Krasser. *Order Types of Point Sets in the Plane*. PhD thesis, Institute for Theoretical Computer Science, Graz University of Technology, Austria, October 2003. One citation on page 2.

[51] M.-J. Lai and L. L. Schumaker. Scattered data interpolation using c2 supersplines of degree six. *SIAM J. Numer. Anal.*, 34(3):905–921, 1997. One citation on page 17.

[52] T. Leighton. Notes on better master theorems for divide-and-conquer recurrences. Technical report, Massachusetts Institute of Technology, 1996. One citation on page 110.

[53] E. L. Lloyd. On triangulations of a set of points in the plane. In *FOCS*, pages 228–240. IEEE Computer Society, 1977. 2 citations on pages 14 and 112.

[54] L. McShine and P. Tetali. On the mixing time of the triangulation walk and other catalan structures. In P. Pardalos and S. Rajasekaran, editors, *Randomization Methods in Algorithm Design*, volume 43 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 147–160. American Mathematical Society, 1998. One citation on page 88.

[55] K. Mehlhorn and P. Sanders. *Algorithms and Data Structures: The Basic Toolbox*. Springer, 2008. One citation on page 15.

[56] K. Mehlhorn and S. Schirra. Exact computation with leda_real - theory and geometric applications. In G. Alefeld, J. Rohn, S. M. Rump, and T. Yamamoto, editors, *Symbolic Algebraic Methods and Verification Methods*, pages 163–172. Springer, 2001. One citation on page 15.

[57] G. L. Miller. Finding small simple cycle separators for 2-connected planar graphs. *JCSS*, 32(3):265–279, June 1986. invited publication. One citation on page 107.

[58] M. S. Molloy, B. Reed, and W. Steiger. On the mixing rate of the triangulation walk. In P. Pardalos and S. Rajasekaran, editors, *Randomization Methods in Algorithm Design*, volume 43 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 179–190. American Mathematical Society, 1998. One citation on page 88.

[59] M. Müller-Hannemann and S. Schirra, editors. *Algorithm Engineering: Bridging the Gap between Algorithm Theory and Practice [outcome of a Dagstuhl Seminar]*, volume 5971 of *Lecture Notes in Computer Science*. Springer, 2010. One citation on page 15.

[60] The on-line encyclopedia of integer sequences. `http://oeis.org/A064062`. One citation on page 74.

[61] A. G. Olaverri, M. Noy, and J. Tejel. Lower bounds on the number of crossing-free subgraphs of k_n. *Comput. Geom.*, 16(4):211–221, 2000. One citation on page 91.

[62] J. O'Rourke. *Art gallery theorems and algorithms*. Oxford University Press, Inc., New York, NY, USA, 1987. One citation on page 81.

[63] J. Pach and P. K. Agarwal. *Combinatorial geometry*. Wiley-Interscience series in discrete mathematics and optimization. Wiley, 1995. One citation on page 4.

[64] D. Pengelley, I. Pivkina, D. Ranjan, and K. Villaverde. A project in algorithms based on a primary historical source about catalan numbers. In D. Baldwin, P. T. Tymann, S. M. Haller, and I. Russell, editors, *SIGCSE*, pages 318–322. ACM, 2006. One citation on page 10.

[65] A. Pilz. Parity properties of geometric graphs. Master's thesis, Graz University of Technology, 2009. One citation on page 44.

[66] M. Pocchiola and G. Vegter. Topologically sweeping visibility complexes via pseudotriangulations. *Discrete & Computational Geometry*, 16(4):419–453, 1996. One citation on page 53.

[67] F. P. Preparata and M. I. Shamos. *Computational Geometry - An Introduction*. Springer, 1985. 2 citations on pages 4 and 15.

[68] S. Ramaswami, P. A. Ramos, and G. T. Toussaint. Converting triangulations to quadrangulations. *Comput. Geom.*, 9(4):257–276, 1998. One citation on page 17.

[69] D. Randall, G. Rote, F. Santos, and J. Snoeyink. Counting triangulations and pseudo-triangulations of wheels. In *CCCG*, pages 149–152, 2001. 2 citations on pages 54 and 57.

[70] S. Ray and R. Seidel. A simple and less slow method for counting triangulations and for related problems. In *EuroCG*, 2004. 10 citations on pages 11, 14, 55, 99, 104, 105, 119, 120, 121, and 122.

[71] A. Razen and E. Welzl. Counting plane graphs with exponential speed-up. In C. S. Calude, G. Rozenberg, and A. Salomaa, editors, *Rainbow of Computer Science*, volume 6570 of *Lecture Notes in Computer Science*, pages 36–46. Springer, 2011. One citation on page 55.

[72] F. Santos and R. Seidel. A better upper bound on the number of triangulations of a planar point set. *J. Comb. Theory, Ser. A*, 102(1):186–193, 2003. 2 citations on pages 3 and 104.

[73] T. Schiffer, F. Aurenhammer, and M. Demuth. Computing convex quadrangulations. *Discrete Applied Mathematics*, 160(4-5):648–656, 2012. 2 citations on pages 17 and 18.

[74] A. Schulz. The existence of a pseudo-triangulation in a given geometric graph. In *EuroCG*, 2006. 2 citations on pages 14 and 112.

[75] M. Sharir and A. Sheffer. Counting triangulations of planar point sets. *Electr. J. Comb.*, 18(1), 2011. 2 citations on pages 3 and 54.

[76] M. Sharir, A. Sheffer, and E. Welzl. Counting plane graphs: Perfect matchings, spanning cycles, and kasteleyn's technique. *CoRR*, abs/1109.5596, 2011. One citation on page 91.

[77] M. Sharir, A. Sheffer, and E. Welzl. On degrees in random triangulations of point sets. *J. Comb. Theory, Ser. A*, 118(7):1979–1999, 2011. 2 citations on pages 3 and 54.

[78] A. Sheffer. Numbers of plane graphs. `http://www.cs.tau.ac.il/~sheffera/counting/PlaneGraphs.html`. One citation on page 91.

[79] R. P. Stanley. *Enumerative Combinatorics: Volume 2*. Number 62 in Cambridge Studies in Advanced Mathematics. Cambridge University Press, 1st edition edition, 2001. One citation on page 10.

[80] R. Steinberg. The state of the three color problem. In J. W. K. John Gimbel and L. V. Quintas, editors, *Quo Vadis, Graph Theory? A Source Book for Challenges and Directions*, volume 55 of *Annals of Discrete Mathematics*, pages 211 – 248. Elsevier, 1993. One citation on page 30.

[81] I. Streinu. A combinatorial approach to planar non-colliding robot arm motion planning. In *FOCS*, pages 443–453. IEEE Computer Society, 2000. 4 citations on pages 53 and 54.

[82] G. T. Toussaint. Quadrangulations of planar sets. In S. G. Akl, F. K. H. A. Dehne, J.-R. Sack, and N. Santoro, editors, *WADS*, volume 955 of *Lecture Notes in Computer Science*, pages 218–227. Springer, 1995. One citation on page 18.

[83] H. S. Wilf. *generatingfunctionology: Third Edition*. CRC Press, 2005. One citation on page 74.